ALIBABA CLOUD

阿里云

消息队列Kafka版

生态对接

文档版本：20220511

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.生态对接概述

本文介绍消息队列Kafka版支持对接的生态。

阿里云生态

云数据库HBase版

- 基于HBase和Spark的数据处理平台

阿里云实时计算Flink

- 基于Flink的资讯场景实时数仓

大数据计算服务MaxCompute

- 接入MaxCompute

数据传输服务DTS

- 使用DTS同步MySQL

容器服务Kubernetes版

- 在Knative上实现Kafka消息推送

阿里云Elasticsearch

- 接入阿里云Elasticsearch

开源生态

h

- 接入Logstash

Filebeat

- 接入Filebeat

MySQL

- 使用Kafka Connect同步MySQL

SQL Server

- 使用Kafka Connect同步SQL Server

2.阿里云生态

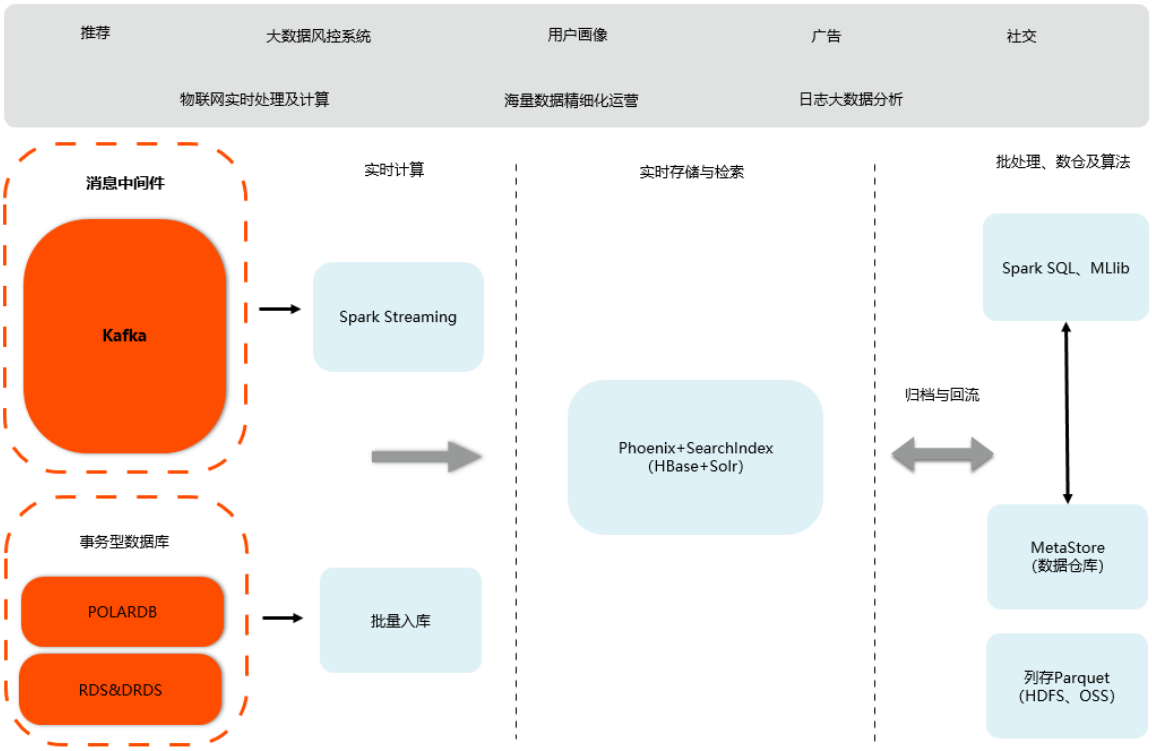
2.1. 搭配云HBase和Spark构建一体化数据处理平台

云HBase X-Pack是基于Apache HBase、Phoenix、Spark深度扩展，融合Solr检索等技术，支持海量数据的一站式存储、检索与分析。融合云Kafka+云HBase X-Pack能够构建一体化的数据处理平台，支持风控、推荐、检索、画像、社交、物联网、时空、表单查询、离线数仓等场景，助力企业数据智能化。

方案架构

下图是业界广泛应用的大数据中台架构。

> **说明** 其中HBase和Spark选择云HBase X-Pack。详情请参见X-pack Spark分析引擎。



- 消息流入：Flume、Logstash或者在线库的Binlog流入消息队列Kafka版。
- 实时计算：通过X-Pack Spark Streaming实时地消费消息队列Kafka版的消息，写入到云HBase中对外提供在线查询。
- 实时存储与检索：云HBase融合Solr以及Phoenix SQL层能够提供海量的实时存储，以及在线查询检索。
- 批处理、数仓及算法：在线存储HBase的数据可以自动归档到X-Pack Spark数仓。全量数据沉淀到Spark数仓（HiveMeta），做批处理、算法分析等复杂计算，结果回流到在线库对外提供查询。

更多信息

- 该套方案的实践操作，请参见Spark对接Kafka快速入门。

> 注意 在Spark接入消息队列Kafka版前，创建Topic和Group。具体操作，请参见步骤三：创建资源。

- 公有云HBase和Spark的示例代码，请参见Demo。

2.2. 基于Flink的资讯场景实时数仓

本文介绍如何针对资讯聚合类业务场景搭建基于消息队列Kafka版和实时计算Flink的实时数仓。

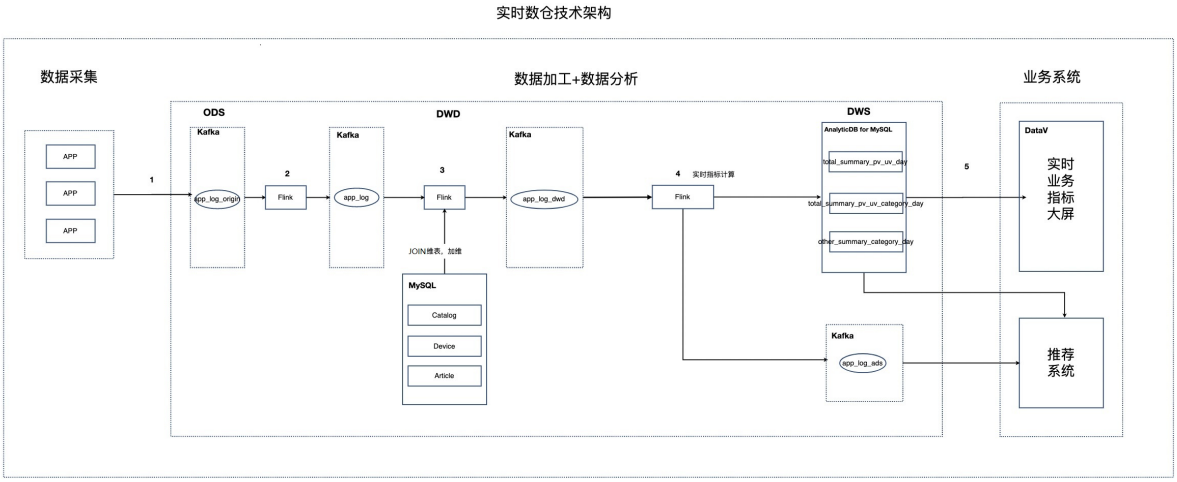
场景描述

本文首先介绍什么是实时数仓以及相关技术架构，接着介绍资讯聚合类业务的典型场景及其业务目标，并据此设计了相应的技术架构。然后介绍如何部署基础环境和搭建实时数仓，并介绍业务系统如何使用实时数仓。

解决的问题

- 通过消息队列Kafka版和实时计算Flink实现实时ETL和数据流。
- 通过消息队列Kafka版和实时计算Flink实现实时数据分析。
- 通过消息队列Kafka版和实时计算Flink实现事件触发。

部署架构图



选用的产品

- **消息队列Kafka版**
  消息队列Kafka版是阿里云基于Apache Kafka构建的高吞吐量、高可扩展性的分布式消息队列服务，广泛用于日志收集、监控数据聚合、流式数据处理、在线和离线分析等，是大数据生态中不可或缺的产品之一，阿里云提供全托管服务，免部署、免运维，更专业、更可靠、更安全。
  更多关于消息队列Kafka版的介绍，参见消息队列Kafka版产品详情页。
- **实时计算**
  实时计算（Alibaba Cloud Realtime Compute）是阿里云提供的基于Apache Flink构建的企业级大数据计算平台。在PB级别的数据集上可以支持亚秒级别的处理延时，赋能用户标准实时数据处理流程和行业解决方案；支持Datastream API作业开发，提供了批流统一的Flink SQL，简化BI场景下的开发；可与用户已使用的大数据组件无缝对接，更多增值特性助力企业实时化转型。
  更多关于实时计算的介绍，参见实时计算产品详情页。
- **DataV数据可视化**

DataV旨在让更多的人看到数据可视化的魅力，帮助非专业的工程师通过图形化的界面轻松搭建专业水准的可视化应用，满足您会议展览、业务监控、风险预警、地理信息分析等多种业务的展示需求。
更多关于阿里云DataV数据可视化的介绍，参见DataV数据可视化产品详情页。

- **专有网络VPC**
  专有网络VPC帮助您基于阿里云构建出一个隔离的网络环境，并可以自定义IP地址范围、网段、路由表和网关等；此外，也可以通过专线、VPN、GRE等连接方式实现云上VPC与传统IDC的互联，构建混合云业务。
  更多关于专有网络VPC的介绍，参见专有网络VPC产品详情页。
- **云数据库RDS**
  阿里云关系型数据库RDS（Relational Database Service）是一种稳定可靠、可弹性伸缩的在线数据库服务。基于阿里云分布式文件系统和SSD盘高性能存储，RDS支持MySQL、SQL Server、PostgreSQL和MariaDB TX引擎，并且提供了容灾、备份、恢复、监控、迁移等方面的全套解决方案，彻底解决数据库运维的烦恼。
  更多关于云数据库RDS的介绍，参见云数据库RDS产品文档。
- **分析型数据库MySQL版**
  分析型数据库MySQL版（AnalyticDB for MySQL）是一种高并发低延时的PB级实时数据仓库，全面兼容MySQL协议以及SQL:2003语法标准，可以毫秒级针对万亿级数据进行即时的多维分析透视和业务探索。
  更多关于分析型数据库MySQL版的介绍，参见分析型数据库MySQL版产品详情页。
- **对象存储OSS**
  阿里云对象存储OSS（Object Storage Service），是阿里云提供的海量、安全、低成本、高可靠的云存储服务。
  更多关于对象存储OSS的介绍，参见对象存储OSS产品详情页。

详细信息

查看最佳实践详情

更多最佳实践

查看更多阿里云最佳实践

2.3. 将消息队列Kafka版的数据迁移至MaxCompute

本文介绍如何使用DataWorks数据同步功能，将消息队列Kafka版集群上的数据迁移至阿里云大数据计算服务MaxCompute，方便您对离线数据进行分析加工。

前提条件

在开始本教程前，确保您在同一地域中已完成以下操作：

- 消息队列Kafka版
  - 购买并部署消息队列Kafka版。具体操作，请参见购买并部署实例。 本文以部署在华东1（杭州）地域（Region）的集群为例。

    > **说明** 消息队列Kafka版实例支持的部署版本（0.10.x版本~2.x版本）、提供的规格类型（标准版和专业版）、支持的网络属性（VPC实例和公网/VPC实例）均支持数据同步。您可以根据业务需要选择。

  - 创建Topic和Group，具体操作，请参见步骤三：创建资源。本文以Topic名称为testkafka，Group名称为console-consumer为例，Group console-consumer将用于消费Topic testkafka中的数据。

- DataWorks
  - 开通DataWorks。
  - 在DataWorks上完成创建业务流程，本文以使用DataWorks简单模式为例。具体操作，请参见创建业务流程。
- MaxCompute
  - 开通MaxCompute。
  - 创建MaxCompute项目。项目信息如下：
    - **模式**：标准模式
    - **项目名称**：生产环境的项目名称默认与DataWorks工作空间的名称一致，可以根据需要修改。开发环境的项目名称默认在DataWorks工作空间的名称增加后缀`_dev`，不可以修改。
    - **访问身份**：生产环境项目的MaxCompute访问身份包括**阿里云主账号**和**阿里云子账号**。**阿里云主账号**即阿里云账号，**阿里云子账号**即RAM用户。开发环境项目的MaxCompute访问身份默认为**任务负责人**，均不可以修改。

    本文以在华东1（杭州）地域创建名为bigdata_DOC的项目为例。

    > **说明** MaxCompute控制台的项目管理和查询编辑功能由DataWorks实现，因此创建MaxCompute项目时，会先创建DataWorks工作空间。MaxCompute项目在MaxCompute控制台**项目管理**页签查看，DataWorks工作空间可以在DataWorks控制台的**工作空间列表**页面查看。

背景信息

大数据计算服务MaxCompute（原ODPS）是一种大数据计算服务，能提供快速、完全托管免运维的EB级云数据仓库解决方案。

DataWorks基于MaxCompute计算和存储，提供工作流可视化开发、调度运维托管的一站式海量数据离线加工分析平台。在数加（一站式大数据平台）中，DataWorks控制台即为MaxCompute控制台。MaxCompute和DataWorks一起向用户提供完善的ETL和数仓管理能力，以及SQL、MR、Graph等多种经典的分布式计算模型，能够更快速地解决用户海量数据计算问题，有效降低企业成本，保障数据安全。

本教程旨在帮助您使用DataWorks，将消息队列Kafka版中的数据导入至MaxCompute，来进一步探索大数据的价值。

步骤一：准备消息队列Kafka版数据

向Topic testkafka中写入数据，以作为迁移至MaxCompute中的数据。由于消息队列Kafka版用于处理流式数据，您可以持续不断地向其中写入数据。为保证测试结果，建议您写入10条以上的数据。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，找到目标Topic，在其**操作**列中，选择**更多 > 体验发送消息**。
6. 在**快速体验消息收发**面板，发送测试消息。
   - **发送方式**选择**控制台**。
     a. 在**消息 Key**文本框中输入消息的Key值，例如demo。
     b. 在**消息内容**文本框输入测试的消息内容，例如 {"key": "test"}。

c. 设置**发送到指定分区**，选择是否指定分区。

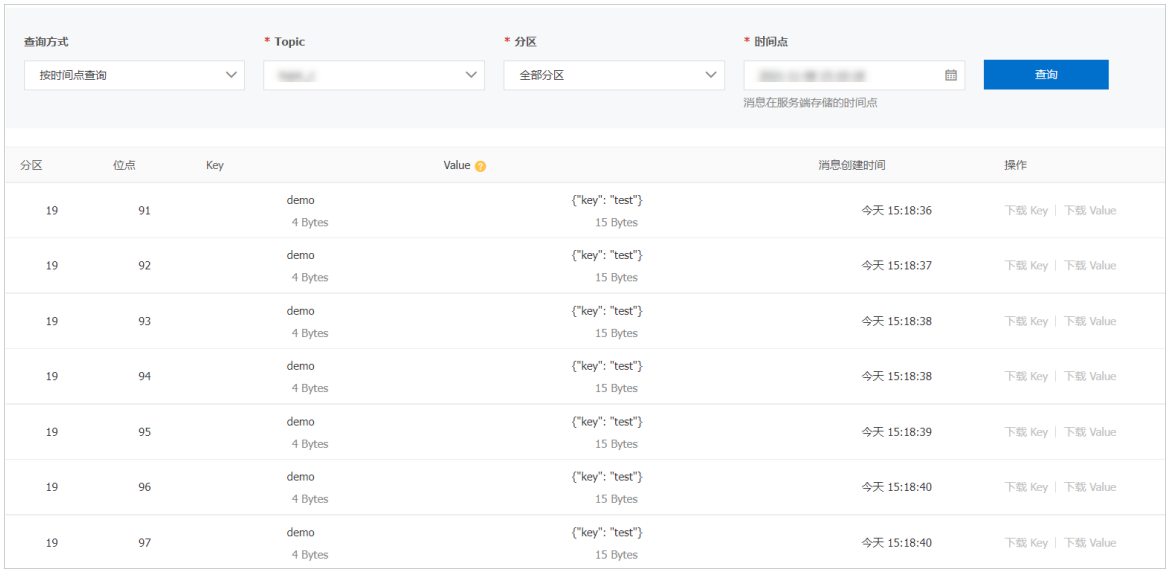
- 单击**是**，在**分区 ID**文本框中输入分区的ID，例如0。如果您需查询分区的ID，请参见查看分区状态。
- 单击**否**，不指定分区。

d. 根据界面提示信息，通过SDK订阅消息，或者执行Docker命令订阅消息。

- **发送方式**选择**Docker**，运行Docker容器。
  a. 执行**运行 Docker 容器生产示例消息**区域的Docker命令，发送消息。
  b. 执行**发送后如何消费消息？**区域的Docker命令，订阅消息。
- **发送方式**选择**SDK**，根据您的业务需求，选择需要的语言或者框架的SDK以及接入方式，通过SDK体验消息收发。

7. 在左侧导航栏，单击**消息查询**，然后在**消息查询**页面，选择查询方式、所属的Topic、分区等信息，单击**查询**，查看之前写入的Topic的数据。

   关于消息查询的更多信息，请参见查询消息。以按时间查询为例，查询的一部分消息如下截图：



步骤二：创建DataWorks表

您需创建DataWorks表，以保证大数据计算服务MaxCompute可以顺利接收消息队列Kafka版数据。为测试便利，本文以使用非分区表为例。

1. 进入**数据开发**页面。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 单击相应工作空间后的**数据开发**。
2. 在**数据开发**页面，右键单击目标业务名称，选择**新建 > MaxCompute > 表**。
3. 在**新建表**页面，选择引擎类型并输入**表名**为testkafka。
4. 在**DDL模式**对话框中，输入如下建表语句，单击**生成表结构**。

```
CREATE TABLE testkafka
(
 key             string,
 value           string,
 partition1      string,
 timestamp1      string,
 offset          string,
 t123            string,
 event_id        string,
 tag             string
) ;
```

其中的每一列，对应于DataWorks数据集成Kafka Reader的默认列：

- __key__表示消息的key。
- __value__表示消息的完整内容 。
- __partition__表示当前消息所在分区。
- __headers__表示当前消息headers信息。
- __offset__表示当前消息的偏移量。
- __timestamp__表示当前消息的时间戳。

您还可以自主命名，详情参见Kafka Reader。

5. 单击**提交到开发环境**。

   具体信息，请参见管理表。

步骤三：新增数据源

将已经写入数据的消息队列Kafka版添加至DataWorks，作为迁移数据源，并添加MaxCompute作为数据迁移的目标源。

1. 新建独享数据集成资源组。

   由于当前DataWorks的默认资源组无法完美支持Kafka插件，您需要使用独享数据集成资源组完成数据同步。详情请参见新增和使用独享数据集成资源组。

2. 登录DataWorks控制台。
3. 在左侧导航栏，单击**工作空间列表**。
4. 在目标工作空间所在行，单击**配置工作空间**，单击左侧导航栏中的**数据源管理**，即可进入**数据源管理**页面。
5. 单击页面右上角的**新增数据源**，即可新增相应的数据源。具体操作，请参见数据源配置。
   - 新增数据源消息队列Kafka版
     a. 在**新增数据源**面板，选择Kafka。

b. 填写数据源Kafka信息。
   - 数据源类型：选择*阿里云实例模式*。
   - 数据源名称：输入新增的数据源名称。
   - 适用环境：选择*开发*。
   - 地区：选择*华东1（杭州）*。
   - 实例ID：在消息队列Kafka版控制台创建的实例ID。
   - 特殊认证方式：保持默认。
   - 资源组连通性：选择*数据集成*。
   - 在独享集群集成资源组列表，目标资源组所在行，单击**测试连通性**。

c. 测试成功后，单击**完成**。

- 新增数据源MaxCompute

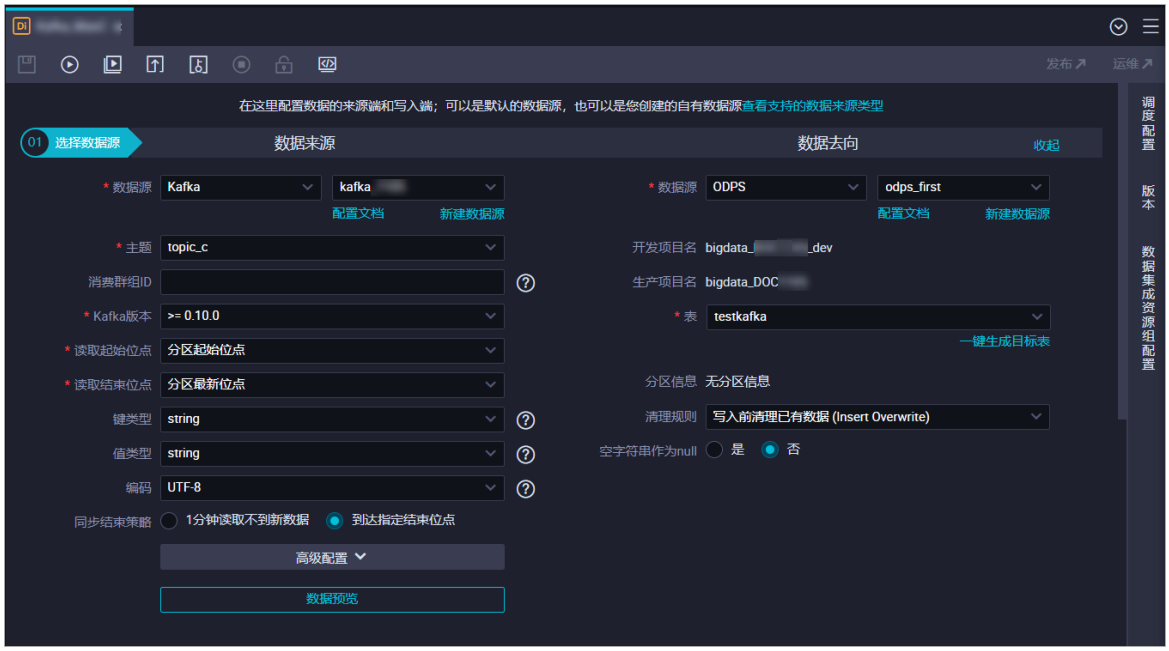
a. 在**新增数据源**面板，选择MaxCompute。

b. 填写数据源MaxCompute信息。
   - 数据源名称：输入新增的数据源名称。
   - 适用环境：选择*开发*。
   - ODPS Endpoint：保持默认。
   - ODPS项目名称：输入ODPS项目名称为bigdata_DOC。
   - AccessKey ID：MaxCompute访问用户的AccessKey ID。更多信息，请参见获取AccessKey。
   - AccessKey Secret：MaxCompute访问用户的AccessKey ID的密码。更多信息，请参见获取AccessKey。
   - 资源组连通性：选择*数据集成*。
   - 在独享集群集成资源组列表，目标资源组所在行，单击**测试连通性**。

c. 测试成功后，单击**完成**。

步骤四：同步数据

1. 进入**数据开发**页面。
   i. 登录DataWorks控制台。
   ii. 在左侧导航栏，单击**工作空间列表**。
   iii. 单击相应工作空间后的**数据开发**。
2. 在**数据开发**页面，右键单击*业务名称*，选择**新建 > 数据集成 > 离线同步**。
3. 在**新建节点**对话框，输入节点名称（即数据同步任务名称），然后单击**提交**。
4. 在创建的节点页面，选择数据源信息。



- 数据来源：选择**数据源**为**Kafka**和步骤三：新增数据源新增的数据源Kafka名称。**主题**选择写入数据的Topic。
- 数据去向：选择**数据去向**的**数据源**为**ODPS**和步骤三：新增数据源新增的数据源MaxCompute名称。**表**选择您在步骤二：创建DataWorks表中创建的表。

您也可以单击配置区域上方的图标，转换为脚本模式，通过脚本配置。示例如下：

```
{
    "type": "job",
    "version": "2.0",
    "steps": [
        {
            "stepType": "kafka",
            "parameter": {
                "server": "localhost:9093",
                "fetchMaxWaitMs": "500",
                "kafkaConfig": {
                    "group.id": "datax_consumer_group"
                },
                "endType": "specific",
                "column": [
                    "__key__",
                    "__value__",
                    "__partition__",
                    "__headers__",
                    "__offset__",
                    "__timestamp__"
                ],
                "timeZone": "Asia/Shanghai",
                "fetchMinBytes": "1",
                "endDateTime": "${endDateTime}",
                "encoding": "UTF-8",
                "version": "10",
                "stopWhenPollEmpty": "false",
```

```
                "beginType": "specific",
                "autoOffsetReset": "none",
                "envType": 0,
                "datasource": "kafka_001",
                "valueType": "string",
                "topic": "topic_c",
                "beginDateTime": "${beginDateTime}",
                "keyType": "string",
                "sessionTimeoutMs": "30000",
                "waitTime": "10"
            },
            "name": "Reader",
            "category": "reader"
        },
        {
            "stepType": "odps",
            "parameter": {
                "partition": "",
                "truncate": true,
                "datasource": "odps_001",
                "envType": 0,
                "column": [
                    "key",
                    "value",
                    "partition1",
                    "timestamp1",
                    "offset",
                    "t123"
                ],
                "emptyAsNull": false,
                "table": "testkafka"
            },
            "name": "Writer",
            "category": "writer"
        }
    ],
    "setting": {
        "executeMode": null,
        "errorLimit": {
            "record": ""
        },
        "speed": {
            "concurrent": 2,
            "throttle": false
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

```
}
```

5. 单击**数据集成资源出配置**，选择步骤三：新增数据源中第一步创建的独享资源组，单击图标，运行任务。

执行结果

完成运行后，**运行日志**中显示运行成功。

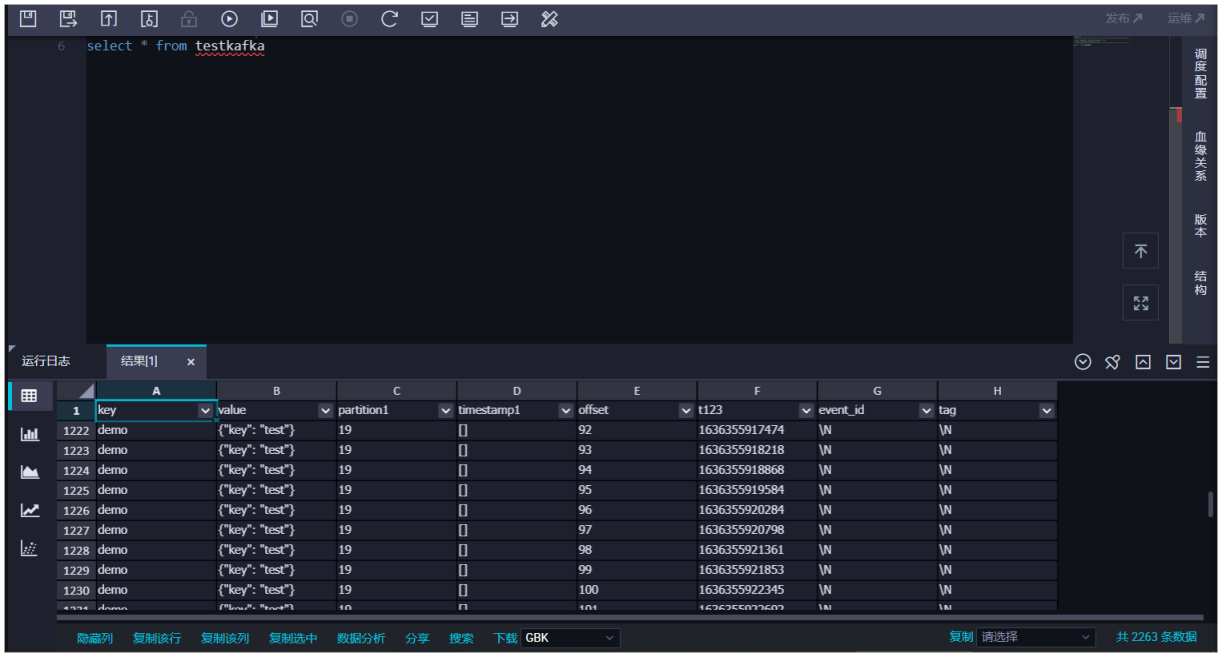
后续步骤

您可以新建一个数据开发任务运行SQL语句，查看当前表中是否已存在从消息队列Kafka版同步过来的数据。本文以 `select * from testkafka` 为例，具体步骤如下：

1. 在左侧导航栏，单击**临时查询**。
2. 在**临时查询**面板，右键单击**临时查询**，选择**新建节点 > ODPS SQL**。
3. 在**新建节点**对话框中，输入**节点名称**，并选择**目标文件夹**。

> **说明** 节点名称的长度不能超过128个字符。

4. 单击**提交**。
5. 在创建的节点页面，输入 `select * from testkafka` ，单击图标，运行完成后，查看运行日志。



2.4. 从自建MySQL同步至

通过数据传输服务DTS（Data Transmission Service），您可以将自建MySQL同步至消息队列Kafka版，扩展消息处理能力。

前提条件

您已完成以下操作：

- 自建MySQL数据库且数据库版本为5.1、5.5、5.6、5.7或8.0版本。
- 购买并部署消息队列Kafka版。详情请参见购买并部署实例。

> **说明** 消息队列Kafka版提供标准版和专业版，两种规格都支持数据同步。您可以根据自建Kafka集群迁移情况选择实例规格，详情请参见评估规格。

- 在目标实例中创建用于接收同步数据的Topic。详情请参见创建Topic。

> **注意**
>
> - Topic名称只能包含字母、数字、下划线（_）和短划线（-）。
> - Topic名称长度限制在3~64字符，长度超过64字符将被自动截取。
> - Topic名称创建后，将无法修改。

背景信息

消息队列Kafka版是阿里云提供的分布式、高吞吐、可扩展的消息队列服务，针对开源的Apache Kafka提供全托管服务，彻底解决开源产品长期以来的痛点，您只需专注于业务开发，无需部署运维。消息队列Kafka版广泛用于日志收集、监控数据聚合、流式数据处理、在线和离线分析等大数据领域，已成为大数据生态中不可或缺的部分。

注意事项

- DTS在执行全量数据初始化时将占用源库和目标库一定的读写资源，可能会导致数据库的负载上升，在数据库性能较差、规格较低或业务量较大的情况下（例如源库有大量慢SQL、存在无主键表或目标库存在死

锁等），可能会加重数据库压力，甚至导致数据库服务不可用。因此您需要在执行数据同步前评估源库和目标库的性能，同时建议您在业务低峰期执行数据同步（例如源库和目标库的CPU负载在30%以下）。

- 如果源数据库没有主键或唯一约束，且所有字段没有唯一性，可能会导致目标数据库中出现重复数据。

功能限制

- 同步对象仅支持数据表，不支持非数据表的对象。
- 不支持自动调整同步对象，如果对同步对象中的数据表进行重命名操作，且重命名后的名称不在同步对象中，那么这部分数据将不再同步到目标消息队列Kafka版集群中。如需将修改后的数据表继续数据同步至目标消息队列Kafka版集群中，您需要进行**修改同步对象**操作，详情请参见新增同步对象。

支持同步的SQL操作

数据传输服务DTS支持同步的SQL操作包括INSERT、UPDATE、DELETE、REPLACE。

消息格式

同步到消息队列Kafka版集群中的数据以avro格式存储，schema定义详情请参见DTS avro schema定义。

> **说明** 数据同步到消息队列Kafka版集群后，您需要根据avro schema定义进行数据解析。

费用说明

数据传输服务DTS费用，请参见产品定价。

准备工作

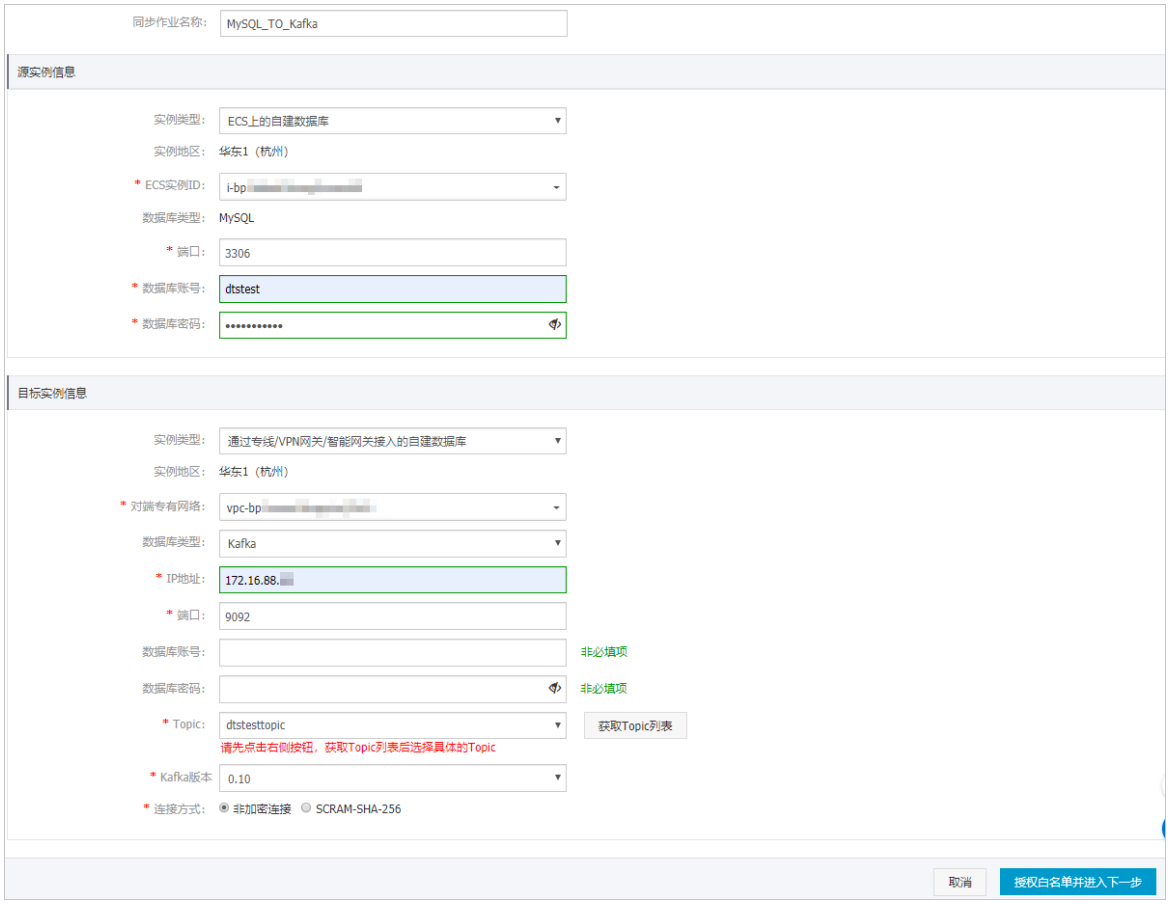
为自建MySQL创建账号并设置binlog

操作步骤

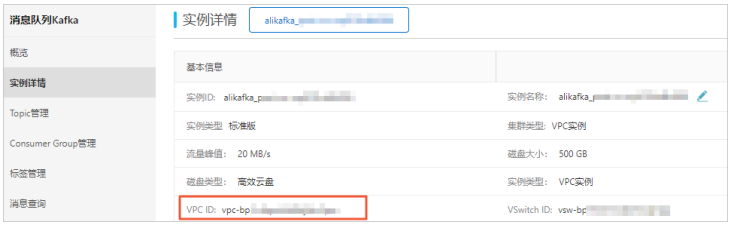
1. 购买数据同步作业，详情请参见购买流程。

   > **说明** 购买时，选择源实例为MySQL，目标实例为Kafka，选择同步拓扑为**单向同步**。

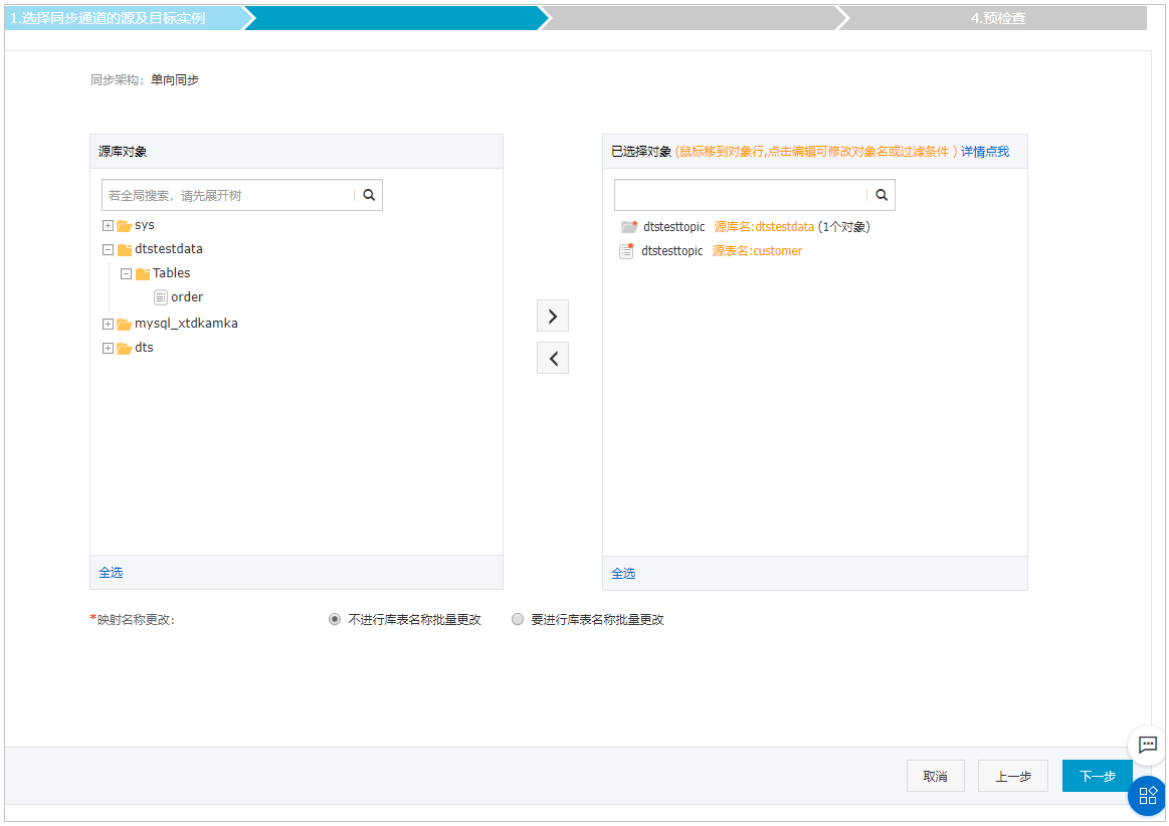
2. 登录数据传输控制台。
3. 在左侧导航栏，单击**数据同步**。
4. 在**同步作业列表**页面顶部，选择同步的目标实例所属地域。
5. 定位至已购买的数据同步实例，单击**配置同步链路**。
6. 配置同步作业的源实例及目标实例信息。



项目	选项	说明
同步作业名称	无	DTS会自动生成一个同步作业名称，建议配置具有业务意义的名称（无唯一性要求），便于后续识别。
源实例信息	实例类型	根据源库部署位置，您可以选择**RDS实例**、**ECS上的自建数据库**或**通过专线、VPN网关、智能网关接入的自建数据库**。 本文以**ECS上的自建数据库**为例介绍配置流程，其他类型的配置流程与该案例类似。
	实例地区	购买数据同步实例时选择的源实例地域信息，不可变更。
	ECS实例ID	选择自建MySQL所属的ECS实例ID。
	数据库类型	固定为**MySQL**，不可变更。
	端口	填入自建MySQL的数据库服务端口。
	数据库账号	填入自建MySQL的数据库账号，该账号需具备REPLICATION SLAVE、REPLICATION CLIENT及所有同步对象的SELECT权限。
	数据库密码	填入该数据库账号对应的密码。

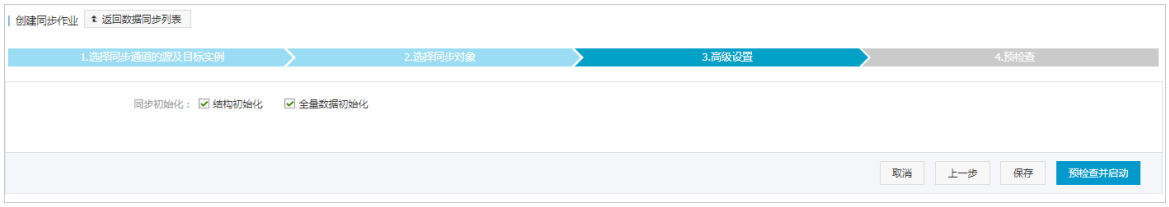
项目	选项	说明
目标实例信息	实例类型	选择**通过专线/VPN网关/智能网关接入的自建数据库**。 **说明** 由于DTS暂时不支持直接选择消息队列Kafka版，此处将其作为自建Kafka来配置数据同步。
	实例地区	购买数据同步实例时选择的目标实例地域信息，不可变更。
	对端专有网络	选择目标消息队列Kafka版实例所属的专有网络ID。您可以在消息队列Kafka版实例的**基本信息**页面中查看到专有网络ID。 
	数据库类型	选择为**Kafka**。
	IP地址	填入消息队列Kafka版实例**默认接入点**中的任意一个IP地址。 **说明** 您可以在消息队列Kafka版实例的**基本信息**页面中，获取**默认接入点**对应的IP地址。
	端口	消息队列Kafka版实例的服务端口，默认为9092。
	数据库账号	填入消息队列Kafka版实例的用户名。 **说明** 如果消息队列Kafka版实例的实例类型为**VPC实例**，无需配置**数据库账号**和**数据库密码**。
	数据库密码	填入该用户名对应的密码。
	Topic	i. 单击右侧的**获取Topic列表**。 ii. 下拉选择具体的Topic名称。
	Kafka版本	根据消息队列Kafka版实例版本，选择对应的版本信息。

7. 单击页面右下角的**授权白名单并进入下一步**。
8. 配置同步对象信息。



配置	说明
同步对象	在**源库对象**区域框中，选择需要同步的对象（仅支持选择数据表），然后单击 › 将其移动到**已选对象**区域框中。 **说明** DTS会自动将表名映射为配置同步的源和目标实例信息时选择的Topic名称。如果需要更换同步的目标Topic（更换后的Topic需是消息队列Kafka版实例中真实存在的），您可以将鼠标指针放置在要进行名称映射的表上，并单击出现的**编辑**进行调整。

9. 上述配置完成后，单击页面右下角的**下一步**。
10. 配置同步初始化的高级配置信息。



说明 同步初始化类型细分为：结构初始化，全量数据初始化。选中**结构初始化**和**全量数据初始化**后，DTS会在增量数据同步之前，将源数据库中待同步对象的结构和存量数据，同步到目标数据库。

11.

12. 在**预检查**对话框中显示**预检查通过**后，关闭**预检查**对话框，同步作业将正式开始。
13. 等待同步作业的链路初始化完成，直至处于**同步中**状态。

    您可以在**数据同步**页面，查看数据同步作业的状态。

2.5. 在Knative上实现Kafka消息推送

Knative已支持Kafka事件源，您可将Knative与消息队列Kafka版对接，在Knative上实现Kafka消息推送。

前提条件

- 一键部署Knative
- 购买并部署消息队列Kafka版实例

  > **注意**
  > - 消息队列Kafka版实例和Knative必须处于同一VPC内。
  > - 消息队列Kafka版实例的版本必须为2.0.0或以上。

- 创建Topic
- 创建Group

背景信息

Knative是一款基于Kubernetes的Serverless框架，其目标是制定云原生、跨平台的Serverless编排标准。Knative主要包括：

- Serving：服务系统，用于配置应用的路由、升级策略、自动扩缩容等。
- Eventing：事件系统，用于自动完成事件的绑定和触发。

要让Eventing（事件系统）正常运行，就必须在Knative集群中实现Channel（内部事件存储层），目前支持的Channel实现方式包括Kafka、NATS。本文以消息队列Kafka版为例介绍如何实现Channel。

适用场景

- 在线短任务处理
- AI音视频消息处理
- 监控告警
- 数据格式转换

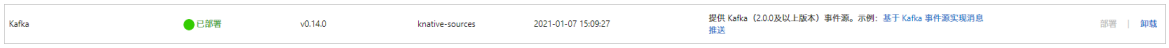
操作流程

在Knative上实现消息队列Kafka版消息推送的操作流程如下图所示。

部署Kafka组件

1. 登录容器服务控制台。

2. 在左侧导航栏，单击**集群**。
3. 在**集群列表**页面，单击要部署Kafka组件的集群的名称。
4. 在**左侧导航栏**，选择**应用 > Knative**。
5. 在**组件管理**页签下的**add-on组件**区域，找到Kafka，在其右侧**操作**列，单击**部署**。
6. 在**部署Kafka**对话框，单击**确定**。
   部署完成后，**Kafka**右侧的**状态**显示**已部署**。



创建event-display服务

1. 在**Knative组件管理**页面，单击**服务管理**页签。
2. 在**Knative服务管理**页面，单击**使用模板创建**。
3. 在**使用模板创建**页面：
   i. 从**集群**列表，选择已部署Knative组件的集群。
   ii. 从**命名空间**列表，选择**default**。
   iii. 从**示例模板**列表，选择**自定义**。
   iv. 在**模板**区域，输入模板信息。

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: event-display
spec:
  template:
    metadata:
      annotations:
        autoscaling.knative.dev/minScale: "1"
    spec:
      containers:
      - image: registry.cn-hangzhou.aliyuncs.com/knative-sample/eventing-sources-cm
d-event_display:bf45b3eb1e7fc4cb63d6a5a6416cf696295484a7662e0cf9ccdf5c080542c21d
```

   v. 单击**创建**。
   vi. 单击**返回**。
   创建完成后，**event-display**右侧的**状态**显示**成功**。

创建kafka-source服务

1. 通过kubectl工具连接集群。
2. 创建KafkaSource服务的配置文件*kafka-source.yaml*。

```
apiVersion: sources.eventing.knative.dev/v1alpha1
kind: KafkaSource
metadata:
  name: kafka-source
spec:
  consumerGroup: demo-topic
  # Broker URL. Replace this with the URLs for your kafka cluster,
  # which is in the format of my-cluster-kafka-bootstrap.my-kafka-namespace:9092.
  bootstrapServers: alikafka-pre-cn-zv**********-1-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-2-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-3-vpc.alikafka.aliyuncs.com:9092
  topics: demo
  sink:
    apiVersion: serving.knative.dev/v1alpha1
    kind: Service
    name: event-display
```

参数	说明	示例值
consumerGroup	创建的Group的名称。	demo-consumer
bootstrapServers	消息队列Kafka版实例的默认接入点。	alikafka-pre-cn-zv**********-1-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-2-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-3-vpc.alikafka.aliyuncs.com:9092
topics	创建的Topic的名称。	demo-topic

3. 执行以下命令创建KafkaSource服务。

```
kubectl apply -f kafka-source.yaml
```

返回示例如下：

```
cb kafkasource.sources.knative.dev/kafka-source created
```

发送消息

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，找到目标Topic，在其**操作**列中，选择**更多 > 体验发送消息**。
6. 在**快速体验消息收发**面板，发送测试消息。
   - **发送方式**选择**控制台**。
     a. 在**消息 Key**文本框中输入消息的Key值，例如demo。
     b. 在**消息内容**文本框输入测试的消息内容，例如 {"key": "test"}。

c. 设置**发送到指定分区**，选择是否指定分区。
   - 单击**是**，在**分区 ID**文本框中输入分区的ID，例如0。如果您需查询分区的ID，请参见查看分区状态。
   - 单击**否**，不指定分区。

d. 根据界面提示信息，通过SDK订阅消息，或者执行Docker命令订阅消息。

- **发送方式**选择**Docker**，运行Docker容器。
  a. 执行**运行 Docker 容器生产示例消息**区域的Docker命令，发送消息。
  b. 执行**发送后如何消费消息？**区域的Docker命令，订阅消息。
- **发送方式**选择**SDK**，根据您的业务需求，选择需要的语言或者框架的SDK以及接入方式，通过SDK体验消息收发。

结果验证

发送消息后，通过 `kubectl logs` 命令查看event-display服务的日志，确认event-display服务已接收到消息队列Kafka版发送的消息。

2.6. 将消息队列Kafka版接入阿里云Elasticsearch

随着时间的积累，消息队列Kafka版中的日志数据会越来越多。当您需要查看并分析庞杂的日志数据时，可通过阿里云Logstash将消息队列Kafka版中的日志数据导入阿里云Elasticsearch，然后通过Kibana进行可视化展示与分析。本文介绍将消息队列Kafka版接入阿里云Elasticsearch的操作方法。

前提条件

在开始本教程前，请确保您已完成以下操作：

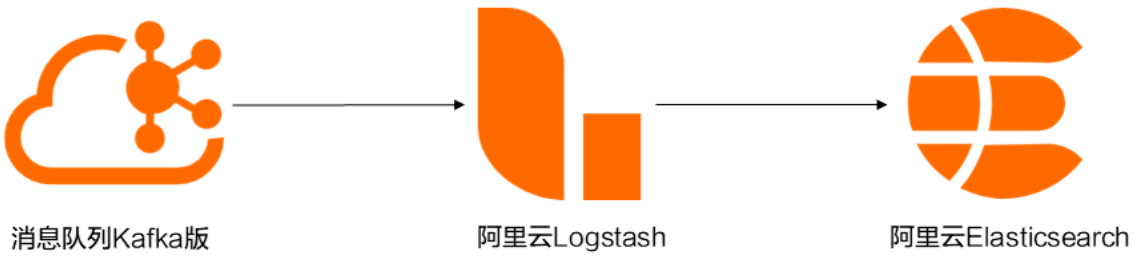
- 购买并部署消息队列Kafka版实例。具体信息，请参见VPC接入。
- 创建阿里云Elasticsearch实例。具体信息，请参见创建阿里云Elasticsearch实例。

> 注意 请注意保存创建阿里云Elasticsearch实例时设置的用户名及密码。该用户名及密码将用于步骤五：创建索引、步骤六：创建管道和步骤七：搜索数据。

- 创建阿里云Logstash实例。具体信息，请参见创建阿里云Logstash实例。

背景信息

通过阿里云Logstash将数据从消息队列Kafka版导入阿里云Elasticsearch的过程如下图所示。



- 消息队列Kafka版

消息队列Kafka版是阿里云提供的分布式、高吞吐、可扩展的消息队列服务。消息队列Kafka版广泛用于日志收集、监控数据聚合、流式数据处理、在线和离线分析等大数据领域，已成为大数据生态中不可或缺的部分。更多信息，请参见什么是消息队列Kafka版。

- 阿里云Elasticsearch
  Elasticsearch简称ES，是一个基于Lucene的实时分布式的搜索与分析引擎，是遵从Apache开源条款的一款开源产品，是当前主流的企业级搜索引擎。它提供了一个分布式服务，可以使您快速的近乎于准实时的存储、查询和分析超大数据集，通常被用来作为构建复杂查询特性和需求强大应用的基础引擎或技术。阿里云Elasticsearch支持5.5.3、6.3.2、6.7.0、6.8.0和7.4.0版本，并提供了商业插件X-Pack服务，致力于数据分析、数据搜索等场景服务。在开源Elasticsearch的基础上提供企业级权限管控、安全监控告警、自动报表生成等功能。更多信息，请参见什么是阿里云Elasticsearch。
- 阿里云Logstash
  阿里云Logstash作为服务器端的数据处理管道，提供了100%兼容开源的Logstash功能。Logstash能够动态地从多个来源采集数据、转换数据，并且将数据存储到所选择的位置。通过输入、过滤和输出插件，Logstash可以对任何类型的事件加工和转换。更多信息，请参见什么是阿里云Logstash。

步骤一：获取VPC环境接入点

阿里云Logstash通过消息队列Kafka版的接入点与消息队列Kafka版在VPC环境下建立连接。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在**实例详情**页面的**接入点信息**页签，获取实例的VPC环境接入点。

消息队列Kafka版支持以下VPC环境接入点：

- 默认接入点：端口号为9092。
- SASL接入点：端口号为9094。如需使用SASL接入点，请开启ACL。您可以提交工单申请开启ACL。

更多信息，请参见接入点对比。

步骤二：创建Topic

创建用于存储消息的Topic。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。

> **注意** Topic需要在应用程序所在的地域（即所部署的ECS的所在地域）进行创建。Topic不能跨地域使用。例如Topic创建在华北2（北京）这个地域，那么消息生产端和消费端也必须运行在华北2（北京）的ECS。

3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，单击**创建 Topic**。

6. 在**创建 Topic**面板，设置Topic属性，然后单击**确定**。

参数	说明	示例
名称	Topic名称。	demo
描述	Topic的简单描述。	demo test
分区数	Topic的分区数量。	12
存储引擎	Topic消息的存储引擎。 消息队列Kafka版支持以下两种存储引擎。 ◦ **云存储**：底层接入阿里云云盘，具有低时延、高性能、持久性、高可靠等特点，采用分布式3副本机制。实例的**规格类型**为**标准版（高写版）**时，存储引擎只能为**云存储**。 ◦ **Local 存储**：使用原生Kafka的ISR复制算法，采用分布式3副本机制。	**云存储**

参数	说明	示例
消息类型	Topic消息的类型。 ○ **普通消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，可能会造成消息乱序。当**存储引擎**选择**云存储**时，默认选择**普通消息**。 ○ **分区顺序消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，仍然保证分区内按照发送顺序存储。但是会出现部分分区发送消息失败，等到分区恢复后即可恢复正常。当**存储引擎**选择**Local 存储**时，默认选择**分区顺序消息**。	**普通消息**

参数	说明	示例
日志清理策略	Topic日志的清理策略。 当**存储引擎**选择**Local 存储**时，需要配置**日志清理策略**。 消息队列Kafka版支持以下两种日志清理策略。 ○ **Delete**：默认的消息清理策略。在磁盘容量充足的情况下，保留在最长保留时间范围内的消息；在磁盘容量不足时（一般磁盘使用率超过85%视为不足），将提前删除旧消息，以保证服务可用性。 ○ **Compact**：使用Kafka Log Compaction日志清理策略。Log Compaction清理策略保证相同Key的消息，最新的value值一定会被保留。主要适用于系统宕机后恢复状态，系统重启后重新加载缓存等场景。例如，在使用Kafka Connect或Confluent Schema Registry时，需要使用Kafka Compact Topic存储系统状态信息或配置信息。 **注意** Compact Topic一般只用在某些生态组件中，例如Kafka Connect或Confluent Schema Registry，其他情况的消息收发请勿为Topic设置该属性。具体信息，请参见消息队列Kafka版Demo库。	Compact
标签	Topic的标签。	demo

创建完成后，在**Topic 管理**页面的列表中显示已创建的Topic。

步骤三：发送消息

向创建的Topic发送消息。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，找到目标Topic，在其**操作**列中，选择**更多 > 体验发送消息**。
6. 在**快速体验消息收发**面板，发送测试消息。

- **发送方式**选择**控制台**。
  a. 在**消息 Key**文本框中输入消息的Key值，例如demo。
  b. 在**消息内容**文本框输入测试的消息内容，例如 {"key": "test"}。
  c. 设置**发送到指定分区**，选择是否指定分区。
     - 单击**是**，在**分区 ID**文本框中输入分区的ID，例如0。如果您需查询分区的ID，请参见查看分区状态。
     - 单击**否**，不指定分区。
  d. 根据界面提示信息，通过SDK订阅消息，或者执行Docker命令订阅消息。
- **发送方式**选择**Docker**，运行Docker容器。
  a. 执行**运行 Docker 容器生产示例消息**区域的Docker命令，发送消息。
  b. 执行**发送后如何消费消息？**区域的Docker命令，订阅消息。
- **发送方式**选择**SDK**，根据您的业务需求，选择需要的语言或者框架的SDK以及接入方式，通过SDK体验消息收发。

步骤四：创建Group

创建阿里云Elasticsearch所属的Group。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Group 管理**。
5. 在**Group 管理**页面，单击**创建 Group**。
6. 在**创建 Group**面板的**Group ID**文本框输入Group的名称，在**描述**文本框简要描述Group，并给Group添加标签，单击**确定**。
   创建完成后，在**Group 管理**页面的列表中显示已创建的Group。

步骤五：创建索引

通过阿里云Elasticsearch创建索引，接收消息队列Kafka版的数据。

1. 登录阿里云Elasticsearch控制台。
2. 在顶部菜单栏，选择地域。
3. 在**实例列表**页面，单击创建的实例。
4. 在左侧导航栏，单击**可视化控制**。
5. 在**Kibana**区域，单击**进入控制台**。
6. 在Kibana登录页面，输入Username和Password，然后单击**Log in**。

   > **说明**
   > - Username为您创建阿里云Elasticsearch实例时设置的用户名。
   > - Password为您创建阿里云Elasticsearch实例时设置的密码。

7. 在Kibana控制台的左侧导航栏，单击**Dev Tools**。
8. 执行以下命令创建索引。

```
PUT /elastic_test
{}
```

步骤六：创建管道

通过阿里云Logstash创建管道。管道部署后，将源源不断地从消息队列Kafka版导入数据进阿里云Elasticsearch。

1. 登录阿里云Logstash控制台。
2. 在顶部菜单栏，选择地域。
3. 在**实例列表**页面，单击创建的实例。
4. 在左侧导航栏，单击**管道管理**。
5. 在**管道列表**区域，单击**创建管道**。
6. 在**Config配置**中，输入配置。

   配置示例如下。

```
input {
    kafka {
    bootstrap_servers => ["alikafka-pre-cn-zv**********-1-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-2-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-3-vpc.alikafka.aliyuncs.com:9092"]
    group_id => "elastic_group"
    topics => ["elastic_test"]
    consumer_threads => 12
    decorate_events => true
    }
}
output {
    elasticsearch {
    hosts => ["http://es-cn-o40xxxxxxxxxxxwm.elasticsearch.aliyuncs.com:9200"]
    index => "elastic_test"
    password => "XXX"
    user => "elastic"
    }
}
```

input参数说明

参数	描述	示例值
bootstrap_servers	消息队列Kafka版的VPC环境接入点。	alikafka-pre-cn-zv**********-1-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-2-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-3-vpc.alikafka.aliyuncs.com:9092
group_id	Group的名称。	elastic_group
topics	Topic的名称。	elastic_test

参数	描述	示例值
consumer_threads	消费线程数。建议与Topic的分区数保持一致。	12
decorate_events	是否包含消息元数据。默认值为false。	true

output参数说明

参数	描述	示例值
hosts	阿里云Elasticsearch服务的访问地址。您可在阿里云Elasticsearch实例的**基本信息**页面获取。	http://es-cn-o40xxxxxxxxxxxwm.elasticsearch.aliyuncs.com:9200
index	索引的名称。	elastic_test
password	访问阿里云Elasticsearch服务的密码。您在创建阿里云Elasticsearch实例时设置的密码。	XXX
user	访问阿里云Elasticsearch服务的用户名。您在创建阿里云Elasticsearch实例时设置的用户名。	elastic

7. 在**管道参数配置**中，输入配置信息，然后单击**保存并部署**。
8. 在**提示**对话框，单击**确认**。

步骤七：搜索数据

您可以在Kibana控制台搜索通过管道导入阿里云Elasticsearch的数据，确认数据是否导入成功。

1. 登录阿里云Elasticsearch控制台。
2. 在顶部菜单栏，选择地域。
3. 在**实例列表**页面，单击创建的实例。
4. 在左侧导航栏，单击**可视化控制**。
5. 在**Kibana**区域，单击**进入控制台**。
6. 在Kibana登录页面，输入Username和Password，然后单击**Log in**。

> **说明**
> - Username为您创建阿里云Elasticsearch实例时设置的用户名。
> - Password为您创建阿里云Elasticsearch实例时设置的密码。

7. 在Kibana控制台的左侧导航栏，单击**Dev Tools**图标。
8. 执行以下命令搜索数据。

```
GET /elastic_test/_search
{}
```

返回结果如下。


```
{
  "took" : 2,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 2,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "elastic_test",
        "_type" : "_doc",
        "_id" : "[redacted]",
        "_score" : 1.0,
        "_source" : {
          "@version" : "1",
          "message" : "1",
          "@timestamp" : "[redacted]T11:33:44.274Z"
        }
      },
      {
        "_index" : "elastic_test",
        "_type" : "_doc",
        "_id" : "[redacted]",
        "_score" : 1.0,
        "_source" : {
          "@version" : "1",
          "message" : "2",
          "@timestamp" : "[redacted]T11:34:44.504Z"
        }
      }
    ]
  }
}
```

3.开源生态

3.1. Logstash

3.1.1. 接入Logstash

本文介绍如何将消息队列Kafka版接入Logstash。

Logstash

Logstash是开源的服务器端数据处理管道，能够同时从多个数据源采集数据，然后对数据进行转换，并将数据写入指定的存储中。Logstash的数据处理流程如下：

1. 输入：采集各种格式、大小和来源的数据。在实际业务中，数据往往以各种各样的形式分散或集中地存储在多个系统中，Logstash支持多种数据输入方式，可以在同一时间从多种数据源采集数据。Logstash能够以连续的流式传输方式从日志、Web应用、数据存储等采集数据。
2. 过滤：实时解析和转换数据。数据从源传输到目标存储的过程中，Logstash过滤器能够解析各个事件，识别已命名的字段来构建结构，并将它们转换成通用格式，通过更轻松、快速的方式分析数据来实现商业价值。
3. 输出：导出数据。Logstash提供多种数据输出方向，灵活解锁众多下游用例。

更多关于Logstash的介绍，请参见Logstash简介。

接入优势

消息队列Kafka版接入Logstash可以带来以下优势：

- 异步处理：提高运行效率，防止突发流量影响用户体验。
- 应用解耦：当应用上下游中有一方存在异常情况，另一方仍能正常运行。
- 减少开销：减少Logstash的资源开销。

接入方案

消息队列Kafka版支持以下方式接入Logstash：

- VPC
  - 作为Input接入
  - 作为Output接入
- 公网
  - 作为Input接入
  - 作为Output接入

3.1.2. VPC

3.1.2.1. 作为Input接入

消息队列Kafka版可以作为Input接入Logstash。本文说明如何在VPC环境下通过Logstash从消息队列Kafka版消费消息。

前提条件

在开始本教程前，请确保您已完成以下操作：

- 购买并部署消息队列Kafka版实例。具体操作，请参见VPC接入。

- 下载并安装Logstash。具体操作，请参见Download Logstash。
- 下载并安装JDK 8。具体操作，请参见Download JDK 8。

步骤一：获取接入点

Logstash通过消息队列Kafka版的接入点与消息队列Kafka版建立连接。

> **说明** 消息队列Kafka版支持以下VPC环境接入点：
> - 默认接入点：端口号为9092。
> - SASL接入点：端口号为9094。如需使用SASL接入点，请开启ACL。您可以提交工单申请开启ACL。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击作为Input接入Logstash的实例名称。
4. 在**实例详情**页面的**接入点信息**区域，获取实例的接入点。在**配置信息**区域，获取**用户名**与**密码**。

> **说明** 不同接入点的差异，请参见接入点对比。

步骤二：创建Topic

创建用于存储消息的Topic。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。

> **注意** Topic需要在应用程序所在的地域（即所部署的ECS的所在地域）进行创建。Topic不能跨地域使用。例如Topic创建在华北2（北京）这个地域，那么消息生产端和消费端也必须运行在华北2（北京）的ECS。

3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，单击**创建 Topic**。
6. 在**创建 Topic**面板，设置Topic属性，然后单击**确定**。

参数	说明	示例
名称	Topic名称。	demo
描述	Topic的简单描述。	demo test
分区数	Topic的分区数量。	12
存储引擎	Topic消息的存储引擎。 消息队列Kafka版支持以下两种存储引擎。 ○ **云存储**：底层接入阿里云云盘，具有低时延、高性能、持久性、高可靠等特点，采用分布式3副本机制。实例的**规格类型**为**标准版（高写版）**时，存储引擎只能为**云存储**。 ○ **Local 存储**：使用原生Kafka的ISR复制算法，采用分布式3副本机制。	**云存储**

参数	说明	示例
消息类型	Topic消息的类型。 ○ **普通消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，可能会造成消息乱序。当**存储引擎**选择**云存储**时，默认选择**普通消息**。 ○ **分区顺序消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，仍然保证分区内按照发送顺序存储。但是会出现部分分区发送消息失败，等到分区恢复后即可恢复正常。当**存储引擎**选择**Local 存储**时，默认选择**分区顺序消息**。	**普通消息**

参数	说明	示例
日志清理策略	Topic日志的清理策略。 当**存储引擎**选择**Local 存储**时，需要配置**日志清理策略**。 消息队列Kafka版支持以下两种日志清理策略。 ○ **Delete**：默认的消息清理策略。在磁盘容量充足的情况下，保留在最长保留时间范围内的消息；在磁盘容量不足时（一般磁盘使用率超过85%视为不足），将提前删除旧消息，以保证服务可用性。 ○ **Compact**：使用Kafka Log Compaction日志清理策略。Log Compaction清理策略保证相同Key的消息，最新的value值一定会被保留。主要适用于系统宕机后恢复状态，系统重启后重新加载缓存等场景。例如，在使用Kafka Connect或Confluent Schema Registry时，需要使用Kafka Compact Topic存储系统状态信息或配置信息。 **注意** Compact Topic一般只用在某些生态组件中，例如Kafka Connect或Confluent Schema Registry，其他情况的消息收发请勿为Topic设置该属性。具体信息，请参见消息队列Kafka版Demo库。	**Compact**
标签	Topic的标签。	demo

创建完成后，在**Topic 管理**页面的列表中显示已创建的Topic。

步骤三：发送消息

向创建的Topic发送消息。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，找到目标Topic，在其**操作**列中，选择**更多 > 体验发送消息**。
6. 在**快速体验消息收发**面板，发送测试消息。

- **发送方式**选择**控制台**。
  a. 在**消息 Key**文本框中输入消息的Key值，例如demo。
  b. 在**消息内容**文本框输入测试的消息内容，例如 {"key": "test"}。
  c. 设置**发送到指定分区**，选择是否指定分区。
     - 单击**是**，在**分区 ID**文本框中输入分区的ID，例如0。如果您需查询分区的ID，请参见查看分区状态。
     - 单击**否**，不指定分区。
  d. 根据界面提示信息，通过SDK订阅消息，或者执行Docker命令订阅消息。
- **发送方式**选择**Docker**，运行Docker容器。
  a. 执行**运行 Docker 容器生产示例消息**区域的Docker命令，发送消息。
  b. 执行**发送后如何消费消息？**区域的Docker命令，订阅消息。
- **发送方式**选择**SDK**，根据您的业务需求，选择需要的语言或者框架的SDK以及接入方式，通过SDK体验消息收发。

步骤四：创建Group

创建Logstash所属的Group。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Group 管理**。
5. 在**Group 管理**页面，单击**创建 Group**。
6. 在**创建 Group**面板的**Group ID**文本框输入Group的名称，在**描述**文本框简要描述Group，并给Group添加标签，单击**确定**。
   创建完成后，在**Group 管理**页面的列表中显示已创建的Group。

步骤五：Logstash消费消息

在安装了Logstash的机器上启动Logstash，从创建的Topic中消费消息。

1. 执行**cd**命令切换到logstash的*bin*目录。
2. 创建*input.conf*配置文件。
   i. 执行命令 `vim input.conf` 创建空的配置文件。
   ii. 按i键进入插入模式。

iii. 输入以下内容。

```
input {
 kafka {
     bootstrap_servers => "alikafka-pre-cn-zv**********-1-vpc.alikafka.aliyuncs.com
:9092,alikafka-pre-cn-zv**********-2-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn
-zv**********-3-vpc.alikafka.aliyuncs.com:9092"
     group_id => "logstash_group"
     topics => ["logstash_test"]
     consumer_threads => 12
     auto_offset_reset => "earliest"
 }
}
output {
 stdout{codec=>rubydebug}
}
```

参数	描述	示例值
bootstrap_servers	消息队列Kafka版提供以下VPC接入点： ■ 默认接入点 ■ SASL接入点	alikafka-pre-cn-zv**********-1-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-2-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-3-vpc.alikafka.aliyuncs.com:9092
group_id	Cosumer Group的名称。	logstash_group
topics	Topic的名称。	logstash_test
consumer_threads	消费线程数。建议与Topic的分区数保持一致。	12
auto_offset_reset	重置偏移量。取值： ■ earliest：读取最早的消息。 ■ latest：读取最新的消息。	earliest

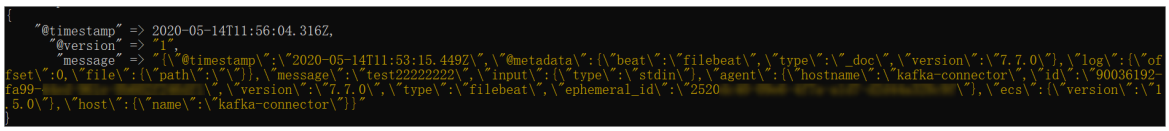
iv. 按*Esc*键回到命令行模式。

v. 按*：*键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

3. 执行以下命令消费消息。

```
./logstash -f input.conf
```

返回结果如下。



更多信息

更多参数设置，请参见Kafka input plugin。

3.1.2.2. 作为Output接入

消息队列Kafka版可以作为Output接入Logstash。本文说明如何在VPC环境下通过Logstash向消息队列Kafka版发送消息。

前提条件

- 购买并部署消息队列Kafka版实例。具体操作，请参见VPC接入。
- 下载并安装Logstash。具体操作，请参见Download Logstash。
- 下载并安装JDK 8。具体操作，请参见Download JDK 8。

步骤一：获取接入点

Logstash通过消息队列Kafka版的接入点与消息队列Kafka版建立连接。

> **说明** 消息队列Kafka版支持以下VPC环境接入点：
>
> - 默认接入点：端口号为9092。
> - SASL接入点：端口号为9094。如需使用SASL接入点，请开启ACL。您可以提交工单申请开启ACL。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击作为Output接入Logstash的实例。
4. 在**实例详情**页面的**接入点信息**区域，获取实例的接入点。在**配置信息**区域，获取**用户名**与**密码**。

> **说明** 不同接入点的差异，请参见接入点对比。

步骤二：创建Topic

创建用于存储消息的Topic。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。

> **注意** Topic需要在应用程序所在的地域（即所部署的ECS的所在地域）进行创建。Topic不能跨地域使用。例如Topic创建在华北2（北京）这个地域，那么消息生产端和消费端也必须运行在华北2（北京）的ECS。

3. 在**实例列表**页面，单击目标实例名称。

4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，单击**创建 Topic**。
6. 在**创建 Topic**面板，设置Topic属性，然后单击**确定**。

参数	说明	示例
名称	Topic名称。	demo
描述	Topic的简单描述。	demo test
分区数	Topic的分区数量。	12
存储引擎	Topic消息的存储引擎。 消息队列Kafka版支持以下两种存储引擎。 ○ **云存储**：底层接入阿里云云盘，具有低时延、高性能、持久性、高可靠等特点，采用分布式3副本机制。实例的**规格类型**为**标准版（高写版）**时，存储引擎只能为**云存储**。 ○ **Local 存储**：使用原生Kafka的ISR复制算法，采用分布式3副本机制。	**云存储**

参数	说明	示例
消息类型	Topic消息的类型。 ◦ **普通消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，可能会造成消息乱序。当**存储引擎**选择**云存储**时，默认选择**普通消息**。 ◦ **分区顺序消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，仍然保证分区内按照发送顺序存储。但是会出现部分分区发送消息失败，等到分区恢复后即可恢复正常。当**存储引擎**选择**Local 存储**时，默认选择**分区顺序消息**。	**普通消息**

参数	说明	示例
日志清理策略	Topic日志的清理策略。 当**存储引擎**选择**Local 存储**时，需要配置**日志清理策略**。 消息队列Kafka版支持以下两种日志清理策略。 ○ **Delete**：默认的消息清理策略。在磁盘容量充足的情况下，保留在最长保留时间范围内的消息；在磁盘容量不足时（一般磁盘使用率超过85%视为不足），将提前删除旧消息，以保证服务可用性。 ○ **Compact**：使用Kafka Log Compaction日志清理策略。Log Compaction清理策略保证相同Key的消息，最新的value值一定会被保留。主要适用于系统宕机后恢复状态，系统重启后重新加载缓存等场景。例如，在使用Kafka Connect或Confluent Schema Registry时，需要使用Kafka Compact Topic存储系统状态信息或配置信息。 **注意** Compact Topic一般只用在某些生态组件中，例如Kafka Connect或Confluent Schema Registry，其他情况的消息收发请勿为Topic设置该属性。具体信息，请参见消息队列Kafka版Demo库。	**Compact**
标签	Topic的标签。	demo

创建完成后，在**Topic 管理**页面的列表中显示已创建的Topic。

步骤三：Logstash发送消息

在安装了Logstash的机器上启动Logstash，向创建的Topic发送消息。

1. 执行**cd**命令切换到logstash的*bin*目录。
2. 创建*output.conf*配置文件。
   i. 执行命令`vim output.conf`创建空的配置文件。
   ii. 按键进入插入模式。

iii. 输入以下内容。

```
input {
    input {
      stdin{}
  }
}
output {
   kafka {
        bootstrap_servers => "alikafka-pre-cn-zv**********-1-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-2-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-3-vpc.alikafka.aliyuncs.com:9092"
        topic_id => "logstash_test"
       }
}
```

参数	描述	示例值
bootstrap_servers	消息队列Kafka版提供以下VPC接入点： ■ 默认接入点 ■ SASL接入点	alikafka-pre-cn-zv**********-1-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-2-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-3-vpc.alikafka.aliyuncs.com:9092
topic_id	Topic的名称。	logstash_test

iv. 按*Esc*键回到命令行模式。

v. 按*：*键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

3. 向创建的Topic发送消息。

i. 执行 `./logstash -f output.conf` 。

ii. 输入*test*，然后按回车键。
返回结果如下。

```
[2020-05-15T14:26:13,576][INFO ][logstash.javapipeline    ][main] Starting pipeline {:pipeline_id=>"main", "pipeline.workers"=>2, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>50, "pipeline.max_inflight"=>250, "pipeline.sources"=>["/home/logstash-7.6.2/bin/output.conf"], :thread=>"#<Thread:0x7be16bf0 run>"}
[2020-05-15T14:26:13,974][INFO ][org.apache.kafka.clients.Metadata][main] [Producer clientId=producer-1] Cluster ID: TsUt5To9Q4-f9YX_5yz2BQ
[2020-05-15T14:26:15,241][INFO ][logstash.javapipeline    ][main] Pipeline started {"pipeline.id"=>"main"}
The stdin plugin is now waiting for input:
[2020-05-15T14:26:15,408][INFO ][logstash.agent           ] Pipelines running {:count=>1, :running_pipelines=>[:main], :non_running_pipelines=>[]}
[2020-05-15T14:26:15,801][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9601}
test
```

步骤四：查看Topic分区

查看消息发送到Topic的情况。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击作为Output接入Logstash的实例。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，找到目标Topic，在其**操作**列中，选择**更多 > 分区状态**。
分区状态信息

参数	说明
分区ID	该Topic分区的ID号。
最小位点	该Topic在当前分区下的最小消费位点。
最大位点	该Topic在当前分区下的最大消费位点。
最近更新时间	本分区中最近一条消息的存储时间。

步骤五：按位点查询消息

您可以根据发送的消息的分区ID和位点信息查询该消息。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**消息查询**。
5. 在**消息查询**页面的**查询方式**列表中，选择**按位点查询**。
6. 在**Topic**列表中，选择消息所属Topic名称；在**分区**列表中，选择消息所属的分区；在**起始位点**文本框，输入消息所在分区的位点，然后单击**查询**。
   展示该查询位点及以后连续的消息。例如，指定的分区和位点都为“5”，那么返回的结果从位点“5”开始。
   查询结果参数解释

参数	描述
分区	消息的Topic分区。
位点	消息的所在的位点。
Key	消息的键（已强制转化为String类型）。
Value	消息的值（已强制转化为String类型），即消息的具体内容。

参数	描述
消息创建时间	发送消息时，客户端自带的或是您指定的`Producer Record`中的消息创建时间。 **说明** ○ 如果配置了该字段，则按配置值显示。 ○ 如果未配置该字段，则默认取消息发送时的系统时间。 ○ 如果显示值为1970/x/x x:x:x，则说明发送时间配置为0或其他有误的值。 ○ 0.9及以前版本的消息队列Kafka版客户端不支持配置该时间。
操作	○ 单击**下载 Key**：下载消息的键值。 ○ 单击**下载 Value**：下载消息的具体内容。 **注意** ○ 查询到的每条消息在控制台上最多显示1 KB的内容，超过1 KB的部分将自动截断。如需查看完整的消息内容，请下载相应的消息。 ○ 仅专业版支持下载消息。 ○ 下载的消息最大为10 MB。如果消息超过10 MB，则只下载10 MB的内容。

更多信息

更多参数设置，请参见Kafka output plugin。

3.1.3. 公网

3.1.3.1. 作为Input接入

消息队列Kafka版可以作为Input接入Logstash。本文说明如何在公网环境下通过Logstash从消息队列Kafka版消费消息。

前提条件

在开始本教程前，请确保您已完成以下操作：

- 购买并部署消息队列Kafka版实例。具体操作，请参见公网+VPC接入。
- 下载并安装Logstash。具体操作，请参见Download Logstash。
- 下载并安装JDK 8。具体操作，请参见Download JDK 8。

步骤一：获取接入信息

Logstash通过消息队列Kafka版的接入点与消息队列Kafka版建立连接，并通过用户名及密码进行校验。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击作为Input接入Logstash的实例名称。
4. 在**实例详情**页面的**接入点信息**区域，获取实例的接入点。在**配置信息**区域，获取**用户名**与**密码**。

> **说明** 不同接入点的差异，请参见接入点对比。

步骤二：创建Topic

创建用于存储消息的Topic。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。

> **注意** Topic需要在应用程序所在的地域（即所部署的ECS的所在地域）进行创建。Topic不能跨地域使用。例如Topic创建在华北2（北京）这个地域，那么消息生产端和消费端也必须运行在华北2（北京）的ECS。

3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，单击**创建 Topic**。
6. 在**创建 Topic**面板，设置Topic属性，然后单击**确定**。

参数	说明	示例
名称	Topic名称。	demo
描述	Topic的简单描述。	demo test
分区数	Topic的分区数量。	12
存储引擎	Topic消息的存储引擎。 消息队列Kafka版支持以下两种存储引擎。 ○ **云存储**：底层接入阿里云云盘，具有低时延、高性能、持久性、高可靠等特点，采用分布式3副本机制。实例的**规格类型**为**标准版（高写版）**时，存储引擎只能为**云存储**。 ○ **Local 存储**：使用原生Kafka的ISR复制算法，采用分布式3副本机制。	**云存储**

参数	说明	示例
消息类型	Topic消息的类型。 ◦ **普通消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，可能会造成消息乱序。当**存储引擎**选择**云存储**时，默认选择**普通消息**。 ◦ **分区顺序消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，仍然保证分区内按照发送顺序存储。但是会出现部分分区发送消息失败，等到分区恢复后即可恢复正常。当**存储引擎**选择**Local 存储**时，默认选择**分区顺序消息**。	**普通消息**

参数	说明	示例
日志清理策略	Topic日志的清理策略。 当**存储引擎**选择**Local 存储**时，需要配置**日志清理策略**。 消息队列Kafka版支持以下两种日志清理策略。 ○ **Delete**：默认的消息清理策略。在磁盘容量充足的情况下，保留在最长保留时间范围内的消息；在磁盘容量不足时（一般磁盘使用率超过85%视为不足），将提前删除旧消息，以保证服务可用性。 ○ **Compact**：使用Kafka Log Compaction日志清理策略。Log Compaction清理策略保证相同Key的消息，最新的value值一定会被保留。主要适用于系统宕机后恢复状态，系统重启后重新加载缓存等场景。例如，在使用Kafka Connect或Confluent Schema Registry时，需要使用Kafka Compact Topic存储系统状态信息或配置信息。 **注意** Compact Topic一般只用在某些生态组件中，例如Kafka Connect或Confluent Schema Registry，其他情况的消息收发请勿为Topic设置该属性。具体信息，请参见消息队列Kafka版Demo库。	Compact
标签	Topic的标签。	demo

创建完成后，在**Topic 管理**页面的列表中显示已创建的Topic。

步骤三：发送消息

向创建的Topic发送消息。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，找到目标Topic，在其**操作**列中，选择**更多 > 体验发送消息**。

6. 在**快速体验消息收发**面板，发送测试消息。
   - **发送方式**选择**控制台**。
     a. 在**消息 Key**文本框中输入消息的Key值，例如demo。
     b. 在**消息内容**文本框输入测试的消息内容，例如 {"key": "test"}。
     c. 设置**发送到指定分区**，选择是否指定分区。
        - 单击**是**，在**分区 ID**文本框中输入分区的ID，例如0。如果您需查询分区的ID，请参见查看分区状态。
        - 单击**否**，不指定分区。
     d. 根据界面提示信息，通过SDK订阅消息，或者执行Docker命令订阅消息。
   - **发送方式**选择**Docker**，运行Docker容器。
     a. 执行**运行 Docker 容器生产示例消息**区域的Docker命令，发送消息。
     b. 执行**发送后如何消费消息？**区域的Docker命令，订阅消息。
   - **发送方式**选择**SDK**，根据您的业务需求，选择需要的语言或者框架的SDK以及接入方式，通过SDK体验消息收发。

步骤四：创建Group

创建Logstash所属的Group。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Group 管理**。
5. 在**Group 管理**页面，单击**创建 Group**。
6. 在**创建 Group**面板的**Group ID**文本框输入Group的名称，在**描述**文本框简要描述Group，并给Group添加标签，单击**确定**。
   创建完成后，在**Group 管理**页面的列表中显示已创建的Group。

步骤五：Logstash消费消息

在安装了Logstash的机器上启动Logstash，从创建的Topic中消费消息。

1. 执行**cd**命令切换到logstash的*bin*目录。
2. 执行以下命令下载*kafka.client.truststore.jks*证书文件。

   ```
   wget https://code.aliyun.com/alikafka/aliware-kafka-demos/raw/master/kafka-log-stash-demo/vpc-ssl/kafka.client.truststore.jks
   ```

3. 创建*jaas.conf*配置文件。
   i. 执行命令 `vim jaas.conf` 创建空的配置文件。
   ii. 按键进入插入模式。

iii. 输入以下内容。

```
KafkaClient {
  org.apache.kafka.common.security.plain.PlainLoginModule required
  username="XXX"
  password="XXX";
};
```

参数	描述	示例值
username	公网/VPC实例的用户名。	alikafka_pre-cn-v0h1***
password	公网/VPC实例的密码。	GQiSmqbQVe3b9hdKLDcIlkrBK6***

iv. 按*Esc*键回到命令行模式。

v. 按*：*键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

4. 创建*input.conf*配置文件。

i. 执行命令`vim input.conf`创建空的配置文件。

ii. 按*i*键进入插入模式。

iii. 输入以下内容。

```
input {
    kafka {
        bootstrap_servers => "alikafka-pre-cn-zv**********-1.alikafka.aliyuncs.com:9093,alikafka-pre-cn-zv**********-2.alikafka.aliyuncs.com:9093,alikafka-pre-cn-zv**********-3.alikafka.aliyuncs.com:9093"
        topics => ["logstash_test"]
        security_protocol => "SASL_SSL"
        sasl_mechanism => "PLAIN"
        jaas_path => "/home/logstash-7.6.2/bin/jaas.conf"
        ssl_truststore_password => "KafkaOnsClient"
        ssl_truststore_location => "/home/logstash-7.6.2/bin/kafka.client.truststore.jks"
        ssl_endpoint_identification_algorithm => ""
        group_id => "logstash_group"
        consumer_threads => 3
        auto_offset_reset => "earliest"
    }
}
output {
    stdout {
        codec => rubydebug
    }
}
```

参数	描述	示例值

参数	描述	示例值
bootstrap_servers	消息队列Kafka版提供的公网接入点为SSL接入点。	alikafka-pre-cn-zv**********-1.alikafka.aliyuncs.com:9093,alikafka-pre-cn-zv**********-2.alikafka.aliyuncs.com:9093,alikafka-pre-cn-zv**********-3.alikafka.aliyuncs.com:9093
topics	Topic的名称。	logstash_test
security_protocol	安全协议。默认为SASL_SSL，无需修改。	SASL_SSL
sasl_mechanism	安全认证机制。默认为PLAIN，无需修改。	PLAIN
jaas_path	*jaas.conf*配置文件位置。	/home/logstash-7.6.2/bin/jaas.conf
ssl_truststore_password	*kafka.client.truststore.jks*证书密码。默认值为KafkaOnsClient，无需修改。	KafkaOnsClient
ssl_truststore_location	*kafka.client.truststore.jks*证书位置。	/home/logstash-7.6.2/bin/kafka.client.truststore.jks
ssl_endpoint_identification_algorithm	6.x及以上版本Logstash需要加上该参数。	空值
group_id	Cosumer Group的名称。	logstash_group
consumer_threads	消费线程数。建议与Topic的分区数保持一致。	3
auto_offset_reset	重置偏移量。取值： ■ earliest：读取最早的消息。 ■ latest：读取最新的消息。	earliest

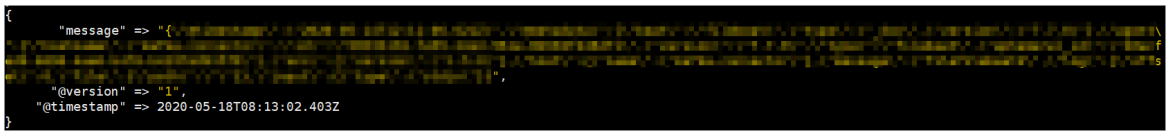
iv. 按*Esc*键回到命令行模式。

v. 按*：*键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

5. 执行以下命令消费消息。

```
./logstash -f input.conf
```

返回结果如下。



更多信息

更多参数设置，请参见Kafka input plugin。

3.1.3.2. 作为Output接入

消息队列Kafka版可以作为Output接入Logstash。本文说明如何在公网环境下通过Logstash向消息队列Kafka版发送消息。

前提条件

在开始本教程前，请确保您已完成以下操作：

- 购买并部署消息队列Kafka版实例。具体操作，请参见公网+VPC接入。
- 下载并安装Logstash。具体操作，请参见Download Logstash。
- 下载并安装JDK 8。具体操作，请参见Download JDK 8。

步骤一：获取接入点

Logstash通过消息队列Kafka版的接入点与消息队列Kafka版建立连接。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击作为Output接入Logstash的实例的名称。
4. 在**实例详情**页面的**接入点信息**区域，获取实例的接入点。在**配置信息**区域，获取**用户名**与**密码**。

> **说明** 不同接入点的差异，请参见接入点对比。

步骤二：创建Topic

创建用于存储消息的Topic。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。

> **注意** Topic需要在应用程序所在的地域（即所部署的ECS的所在地域）进行创建。Topic不能跨地域使用。例如Topic创建在华北2（北京）这个地域，那么消息生产端和消费端也必须运行在华北2（北京）的ECS。

3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，单击**创建 Topic**。
6. 在**创建 Topic**面板，设置Topic属性，然后单击**确定**。

参数	说明	示例
名称	Topic名称。	demo
描述	Topic的简单描述。	demo test
分区数	Topic的分区数量。	12
存储引擎	Topic消息的存储引擎。 消息队列Kafka版支持以下两种存储引擎。 ○ **云存储**：底层接入阿里云云盘，具有低时延、高性能、持久性、高可靠等特点，采用分布式3副本机制。实例的**规格类型**为**标准版（高写版）**时，存储引擎只能为**云存储**。 ○ **Local 存储**：使用原生Kafka的ISR复制算法，采用分布式3副本机制。	**云存储**

参数	说明	示例
消息类型	Topic消息的类型。 ○ **普通消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，可能会造成消息乱序。当**存储引擎**选择**云存储**时，默认选择**普通消息**。 ○ **分区顺序消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，仍然保证分区内按照发送顺序存储。但是会出现部分分区发送消息失败，等到分区恢复后即可恢复正常。当**存储引擎**选择**Local 存储**时，默认选择**分区顺序消息**。	**普通消息**

参数	说明	示例
日志清理策略	Topic日志的清理策略。 当**存储引擎**选择**Local 存储**时，需要配置**日志清理策略**。 消息队列Kafka版支持以下两种日志清理策略。 ○ **Delete**：默认的消息清理策略。在磁盘容量充足的情况下，保留在最长保留时间范围内的消息；在磁盘容量不足时（一般磁盘使用率超过85%视为不足），将提前删除旧消息，以保证服务可用性。 ○ **Compact**：使用Kafka Log Compaction日志清理策略。Log Compaction清理策略保证相同Key的消息，最新的value值一定会被保留。主要适用于系统宕机后恢复状态，系统重启后重新加载缓存等场景。例如，在使用Kafka Connect或Confluent Schema Registry时，需要使用Kafka Compact Topic存储系统状态信息或配置信息。 **注意** Compact Topic一般只用在某些生态组件中，例如Kafka Connect或Confluent Schema Registry，其他情况的消息收发请勿为Topic设置该属性。具体信息，请参见消息队列Kafka版Demo库。	**Compact**
标签	Topic的标签。	demo

创建完成后，在**Topic 管理**页面的列表中显示已创建的Topic。

步骤三：Logstash发送消息

在安装了Logstash的机器上启动Logstash，向创建的Topic发送消息。

1. 执行**cd**命令切换到logstash的*bin*目录。
2. 执行以下命令下载*kafka.client.truststore.jks*证书文件。

```
wget https://code.aliyun.com/alikafka/aliware-kafka-demos/raw/master/kafka-log-stash-demo/vpc-ssl/kafka.client.truststore.jks
```

3. 创建*jaas.conf*配置文件。
   i. 执行命令`vim jaas.conf`创建空的配置文件。

ii. 按i键进入插入模式。

iii. 输入以下内容。

```
KafkaClient {
  org.apache.kafka.common.security.plain.PlainLoginModule required
  username="XXX"
  password="XXX";
};
```

参数	描述	示例值
username	公网/VPC实例的用户名。	alikafka_pre-cn-v0h1***
password	公网/VPC实例的密码。	GQiSmqbQVe3b9hdKLDcIlkrBK6***

iv. 按*Esc*键回到命令行模式。

v. 按*：*键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

4. 创建*output.conf配置文件*。

i. 执行命令`vim output.conf`创建空的配置文件。

ii. 按i键进入插入模式。

iii. 输入以下内容。

```
input {
    stdin{}
}
output {
   kafka {
        bootstrap_servers => "alikafka-pre-cn-zv**********-1.alikafka.aliyuncs.com:9093,alikafka-pre-cn-zv**********-2.alikafka.aliyuncs.com:9093,alikafka-pre-cn-zv**********-3.alikafka.aliyuncs.com:9093"
        topic_id => "logstash_test"
        security_protocol => "SASL_SSL"
        sasl_mechanism => "PLAIN"
        jaas_path => "/home/logstash-7.6.2/bin/jaas.conf"
        ssl_truststore_password => "KafkaOnsClient"
        ssl_truststore_location => "/home/logstash-7.6.2/bin/kafka.client.truststore.jks"
        ssl_endpoint_identification_algorithm => ""
    }
}
```

参数	描述	示例值
bootstrap_servers	消息队列Kafka版提供的公网接入点为SSL接入点。	alikafka-pre-cn-zv**********-1.alikafka.aliyuncs.com:9093,alikafka-pre-cn-zv**********-2.alikafka.aliyuncs.com:9093,alikafka-pre-cn-zv**********-3.alikafka.aliyuncs.com:9093
topic_id	Topic的名称。	logstash_test
security_protocol	安全协议。默认为SASL_SSL，无需修改。	SASL_SSL
sasl_mechanism	安全认证机制。默认为PLAIN，无需修改。	PLAIN
jaas_path	*jaas.conf*配置文件位置。	/home/logstash-7.6.2/bin/jaas.conf
ssl_truststore_password	*kafka.client.truststore.jks*证书密码。默认值为KafkaOnsClient，无需修改。	KafkaOnsClient
ssl_truststore_location	*kafka.client.truststore.jks*证书位置。	/home/logstash-7.6.2/bin/kafka.client.truststore.jks
ssl_endpoint_identification_algorithm	SSL接入点辨识算法。6.x及以上版本Logstash需要加上该参数。	空值

iv. 按*Esc*键回到命令行模式。

v. 按：键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

5. 向创建的Topic发送消息。
   i. 执行 `./logstash -f output.conf` 。
   ii. 输入*test*，然后按回车键。

```
[2020-05-18T17:18:10,170][INFO ][org.apache.kafka.clients.Metadata][main] [Producer clientId=producer-1] Cluster ID: [redacted]
[2020-05-18T17:18:10,789][INFO ][logstash.javapipeline    ][main] Pipeline started {"pipeline.id"=>"main"}
The stdin plugin is now waiting for input:
[2020-05-18T17:18:10,881][INFO ][logstash.agent           ] Pipelines running {:count=>1, :running_pipelines=>[:main], :non_running_pipelines=>[]}
[2020-05-18T17:18:11,127][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>[redacted]}
test
```

步骤四：查看Topic分区

查看消息发送到Topic的情况。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，找到目标Topic，在其**操作**列中，选择**更多 > 分区状态**。

   分区状态信息

参数	说明
分区ID	该Topic分区的ID号。
最小位点	该Topic在当前分区下的最小消费位点。
最大位点	该Topic在当前分区下的最大消费位点。
最近更新时间	本分区中最近一条消息的存储时间。

步骤五：按位点查询消息

您可以根据发送的消息的分区ID和位点信息查询该消息。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**消息查询**。

5. 在**消息查询**页面的**查询方式**列表中，选择**按位点查询**。
6. 在**Topic**列表中，选择消息所属Topic名称；在**分区**列表中，选择消息所属的分区；在**起始位点**文本框，输入消息所在分区的位点，然后单击**查询**。
   展示该查询位点及以后连续的消息。例如，指定的分区和位点都为“5”，那么返回的结果从位点“5”开始。
   查询结果参数解释

参数	描述
分区	消息的Topic分区。
位点	消息的所在的位点。
Key	消息的键（已强制转化为String类型）。
Value	消息的值（已强制转化为String类型），即消息的具体内容。
消息创建时间	发送消息时，客户端自带的或是您指定的`ProducerRecord`中的消息创建时间。 **说明** ○ 如果配置了该字段，则按配置值显示。 ○ 如果未配置该字段，则默认取消息发送时的系统时间。 ○ 如果显示值为1970/x/x x:x:x，则说明发送时间配置为0或其他有误的值。 ○ 0.9及以前版本的消息队列Kafka版客户端不支持配置该时间。
操作	○ 单击**下载 Key**：下载消息的键值。 ○ 单击**下载 Value**：下载消息的具体内容。 **注意** ○ 查询到的每条消息在控制台上最多显示1 KB的内容，超过1 KB的部分将自动截断。如需查看完整的消息内容，请下载相应的消息。 ○ 仅专业版支持下载消息。 ○ 下载的消息最大为10 MB。如果消息超过10 MB，则只下载10 MB的内容。

更多信息

更多参数设置，请参见Kafka output plugin。

3.2. Filebeat

3.2.1. 接入Filebeat

本文介绍如何将消息队列Kafka版接入Filebeat。

Filebeat

Filebeat是用于转发和集中日志数据的轻量级传输程序。Filebeat可以监听指定的日志文件或位置，从中收集日志事件并将其转发到Elasticsearch或Logstash进行索引。Filebeat的工作原理如下：

1. Filebeat启动一个或多个Input，Input在指定的位置中查找日志数据。
2. Filebeat为每个找到的日志启动Harvester，Harvester读取日志并将日志数据发送到libbeat。
3. libbeat聚集数据，然后将聚集的数据发送到配置的Output。

接入优势

消息队列Kafka版接入Filebeat可以带来以下优势：

- 异步处理：防止突发流量。
- 应用解耦：当下游异常时，不会影响上游工作。
- 减少开销：减少Filebeat的资源开销。

接入方案

消息队列Kafka版支持以下方式接入Filebeat：

- VPC
  - 作为Input接入
  - 作为Output接入
- 公网
  - 作为Input接入
  - 作为Output接入

3.2.2. VPC

3.2.2.1. 作为Input接入

消息队列Kafka版可以作为Input接入Filebeat。本文说明如何在VPC环境下通过Filebeat从消息队列Kafka版消费消息。

背景信息

在开始本教程前，请确保您已完成以下操作：

- 购买并部署消息队列Kafka版实例。具体操作，请参见VPC接入。
- 下载并安装Filebeat。具体操作，请参见Download Filebeat。
- 下载并安装JDK 8。具体操作，请参见Download JDK 8。

步骤一：获取接入点

Filebeat通过消息队列Kafka版的接入点与消息队列Kafka版建立连接。

1. 登录消息队列Kafka版控制台。

2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击要作为Input接入Filebeat的实例名称。
4. 在**实例详情**页面的**接入点信息**区域，获取实例的接入点。在**配置信息**区域，获取**用户名**与**密码**。

> **说明** 不同接入点的差异，请参见接入点对比。

步骤二：创建Topic

创建用于存储消息的Topic。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。

> **注意** Topic需要在应用程序所在的地域（即所部署的ECS的所在地域）进行创建。Topic不能跨地域使用。例如Topic创建在华北2（北京）这个地域，那么消息生产端和消费端也必须运行在华北2（北京）的ECS。

3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，单击**创建 Topic**。
6. 在**创建 Topic**面板，设置Topic属性，然后单击**确定**。

参数	说明	示例
名称	Topic名称。	demo
描述	Topic的简单描述。	demo test
分区数	Topic的分区数量。	12
存储引擎	Topic消息的存储引擎。 消息队列Kafka版支持以下两种存储引擎。 ○ **云存储**：底层接入阿里云云盘，具有低时延、高性能、持久性、高可靠等特点，采用分布式3副本机制。实例的**规格类型**为**标准版（高写版）**时，存储引擎只能为**云存储**。 ○ **Local 存储**：使用原生Kafka的ISR复制算法，采用分布式3副本机制。	**云存储**

参数	说明	示例
消息类型	Topic消息的类型。 ○ **普通消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，可能会造成消息乱序。当**存储引擎**选择**云存储**时，默认选择**普通消息**。 ○ **分区顺序消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，仍然保证分区内按照发送顺序存储。但是会出现部分分区发送消息失败，等到分区恢复后即可恢复正常。当**存储引擎**选择**Local 存储**时，默认选择**分区顺序消息**。	**普通消息**

参数	说明	示例
日志清理策略	Topic日志的清理策略。 当**存储引擎**选择**Local 存储**时，需要配置**日志清理策略**。 消息队列Kafka版支持以下两种日志清理策略。 ○ **Delete**：默认的消息清理策略。在磁盘容量充足的情况下，保留在最长保留时间范围内的消息；在磁盘容量不足时（一般磁盘使用率超过85%视为不足），将提前删除旧消息，以保证服务可用性。 ○ **Compact**：使用Kafka Log Compaction日志清理策略。Log Compaction清理策略保证相同Key的消息，最新的value值一定会被保留。主要适用于系统宕机后恢复状态，系统重启后重新加载缓存等场景。例如，在使用Kafka Connect或Confluent Schema Registry时，需要使用Kafka Compact Topic存储系统状态信息或配置信息。 **注意** Compact Topic一般只用在某些生态组件中，例如Kafka Connect或Confluent Schema Registry，其他情况的消息收发请勿为Topic设置该属性。具体信息，请参见消息队列Kafka版Demo库。	**Compact**
标签	Topic的标签。	demo

创建完成后，在**Topic 管理**页面的列表中显示已创建的Topic。

步骤三：发送消息

向创建的Topic发送消息。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。

5. 在**Topic 管理**页面，找到目标Topic，在其**操作**列中，选择**更多 > 体验发送消息**。
6. 在**快速体验消息收发**面板，发送测试消息。
   - **发送方式**选择**控制台**。
     a. 在**消息 Key**文本框中输入消息的Key值，例如demo。
     b. 在**消息内容**文本框输入测试的消息内容，例如 {"key": "test"}。
     c. 设置**发送到指定分区**，选择是否指定分区。
        - 单击**是**，在**分区 ID**文本框中输入分区的ID，例如0。如果您需查询分区的ID，请参见查看分区状态。
        - 单击**否**，不指定分区。
     d. 根据界面提示信息，通过SDK订阅消息，或者执行Docker命令订阅消息。
   - **发送方式**选择**Docker**，运行Docker容器。
     a. 执行**运行 Docker 容器生产示例消息**区域的Docker命令，发送消息。
     b. 执行**发送后如何消费消息？**区域的Docker命令，订阅消息。
   - **发送方式**选择**SDK**，根据您的业务需求，选择需要的语言或者框架的SDK以及接入方式，通过SDK体验消息收发。

步骤四：创建Group

创建Filebeat所属的Group。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Group 管理**。
5. 在**Group 管理**页面，单击**创建 Group**。
6. 在**创建 Group**面板的**Group ID**文本框输入Group的名称，在**描述**文本框简要描述Group，并给Group添加标签，单击**确定**。
   创建完成后，在**Group 管理**页面的列表中显示已创建的Group。

步骤五：Filebeat消费消息

在安装了Filebeat的机器上启动Filebeat，从创建的Topic中消费消息。

1. 执行**cd**命令切换到Filebeat的安装目录。
2. 创建*input.yml*配置文件。
   i. 执行命令 `vim input.yml` 创建空的配置文件。
   ii. 按*i*键进入插入模式。

iii. 输入以下内容。

```
filebeat.inputs:
- type: kafka
  hosts:
    - alikafka-pre-cn-zv**********-1-vpc.alikafka.aliyuncs.com:9092
    - alikafka-pre-cn-zv**********-2-vpc.alikafka.aliyuncs.com:9092
    - alikafka-pre-cn-zv**********-3-vpc.alikafka.aliyuncs.com:9092
  topics: ["filebeat_test"]
  group_id: "filebeat_group"
output.console:
  pretty: true
```

参数	描述	示例值
type	Filebeat的Input类型。	kafka
hosts	消息队列Kafka版提供以下VPC接入点： ■ 默认接入点 ■ SASL接入点	- alikafka-pre-cn-zv**********-1-vpc.alikafka.aliyuncs.com:9092 - alikafka-pre-cn-zv**********-2-vpc.alikafka.aliyuncs.com:9092 - alikafka-pre-cn-zv**********-3-vpc.alikafka.aliyuncs.com:9092
topics	Topic的名称。	filebeat_test
group_id	Group的名称。	filebeat_group

更多参数说明，请参见Kafka input plugin。

iv. 按*Esc*键回到命令行模式。

v. 按*：*键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

3. 执行以下命令消费消息。

```
./filebeat -c ./input.yml
```

3.2.2.2. 作为Output接入

消息队列Kafka版可以作为Output接入Filebeat。本文说明如何在VPC环境下通过Filebeat向消息队列Kafka版发送消息。

前提条件

在开始本教程前，请确保您已完成以下操作：

- 购买并部署消息队列Kafka版实例。更多信息，请参见VPC接入。
- 下载并安装Filebeat。更多信息，请参见Download Filebeat。
- 下载并安装JDK 8。更多信息，请参见Download JDK 8。

步骤一：获取接入点

Filebeat通过消息队列Kafka版的接入点与消息队列Kafka版建立连接。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击作为Output接入Filebeat的实例名称。
4. 在**实例详情**页面的**接入点信息**区域，获取实例的接入点。在**配置信息**区域，获取**用户名**与**密码**。

> **说明** 不同接入点的差异，请参见接入点对比。

步骤二：创建Topic

创建用于存储消息的Topic。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。

> **注意** Topic需要在应用程序所在的地域（即所部署的ECS的所在地域）进行创建。Topic不能跨地域使用。例如Topic创建在华北2（北京）这个地域，那么消息生产端和消费端也必须运行在华北2（北京）的ECS。

3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，单击**创建 Topic**。
6. 在**创建 Topic**面板，设置Topic属性，然后单击**确定**。

参数	说明	示例
名称	Topic名称。	demo
描述	Topic的简单描述。	demo test
分区数	Topic的分区数量。	12

参数	说明	示例
存储引擎	Topic消息的存储引擎。 消息队列Kafka版支持以下两种存储引擎。 ○ **云存储**：底层接入阿里云云盘，具有低时延、高性能、持久性、高可靠等特点，采用分布式3副本机制。实例的**规格类型**为**标准版（高写版）**时，存储引擎只能为**云存储**。 ○ **Local 存储**：使用原生Kafka的ISR复制算法，采用分布式3副本机制。	**云存储**
消息类型	Topic消息的类型。 ○ **普通消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，可能会造成消息乱序。**当存储引擎选择云存储时，默认选择普通消息。** ○ **分区顺序消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，仍然保证分区内按照发送顺序存储。但是会出现部分分区发送消息失败，等到分区恢复后即可恢复正常。**当存储引擎选择Local 存储时，默认选择分区顺序消息。**	**普通消息**

参数	说明	示例
日志清理策略	Topic日志的清理策略。 当**存储引擎**选择**Local 存储**时，需要配置**日志清理策略**。 消息队列Kafka版支持以下两种日志清理策略。 ○ **Delete**：默认的消息清理策略。在磁盘容量充足的情况下，保留在最长保留时间范围内的消息；在磁盘容量不足时（一般磁盘使用率超过85%视为不足），将提前删除旧消息，以保证服务可用性。 ○ **Compact**：使用Kafka Log Compaction日志清理策略。Log Compaction清理策略保证相同Key的消息，最新的value值一定会被保留。主要适用于系统宕机后恢复状态，系统重启后重新加载缓存等场景。例如，在使用Kafka Connect或Confluent Schema Registry时，需要使用Kafka Compact Topic存储系统状态信息或配置信息。 **注意** Compact Topic一般只用在某些生态组件中，例如Kafka Connect或Confluent Schema Registry，其他情况的消息收发请勿为Topic设置该属性。具体信息，请参见消息队列Kafka版Demo库。	**Compact**
标签	Topic的标签。	demo

创建完成后，在**Topic 管理**页面的列表中显示已创建的Topic。

步骤三：Filebeat发送消息

在安装了Filebeat的机器上启动Filebeat，向创建的Topic发送消息。

1. 执行**cd**命令切换到Filebeat的安装目录。
2. 创建*output.conf*配置文件。
   i. 执行命令`vim output.conf`创建空的配置文件。
   ii. 按键进入插入模式。

iii. 输入以下内容。

```
filebeat.inputs:
- type: stdin
output.kafka:
  hosts: ["alikafka-pre-cn-zv**********-1-vpc.alikafka.aliyuncs.com:9092", "alikafk
a-pre-cn-zv**********-2-vpc.alikafka.aliyuncs.com:9092", "alikafka-pre-cn-zv*******
***-3-vpc.alikafka.aliyuncs.com:9092"]
  topic: 'filebeat_test'
  required_acks: 1
  compression: none
  max_message_bytes: 1000000
```

参数	描述	示例值
hosts	消息队列Kafka版提供以下VPC接入点： ■ 默认接入点 ■ SASL接入点	alikafka-pre-cn-zv**********-1-vpc.alikafka.aliyuncs.com:9092，alikafka-pre-cn-zv**********-2-vpc.alikafka.aliyuncs.com:9092，alikafka-pre-cn-zv**********-3-vpc.alikafka.aliyuncs.com:9092
topic	Topic的名称。	filebeat_test
required_acks	ACK可靠性。取值： ■ 0：无响应 ■ 1：等待本地提交 ■ -1：等待所有副本提交 默认值为1。	1
compression	数据压缩编译码器。默认值为gzip。取值： ■ none：无 ■ snappy：用来压缩和解压缩的C++开发包 ■ lz4：着重于压缩和解压缩速度的无损数据压缩算法 ■ gzip：GNU自由软件的文件压缩程序	none
max_message_bytes	最大消息大小。单位为字节。默认值为1000000。该值应小于您配置的消息队列Kafka版最大消息大小。	1000000

更多参数说明，请参见Kafka output plugin。

iv. 按*Esc*键回到命令行模式。

v. 按*：*键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

3. 向创建的Topic发送消息。
   i. 执行 `./filebeat -c ./output.yml` 。
   ii. 输入*test*，然后按回车键。

步骤四：查看Topic分区

查看消息发送到Topic的情况。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，找到目标Topic，在其**操作**列中，选择**更多 > 分区状态**。

   分区状态信息

参数	说明
分区ID	该Topic分区的ID号。
最小位点	该Topic在当前分区下的最小消费位点。
最大位点	该Topic在当前分区下的最大消费位点。
最近更新时间	本分区中最近一条消息的存储时间。

步骤五：按位点查询消息

您可以根据发送的消息的分区ID和位点信息查询该消息。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**消息查询**。
5. 在**消息查询**页面的**查询方式**列表中，选择**按位点查询**。

6. 在**Topic**列表中，选择消息所属Topic名称；在**分区**列表中，选择消息所属的分区；在**起始位点**文本框，输入消息所在分区的位点，然后单击**查询**。
   展示该查询位点及以后连续的消息。例如，指定的分区和位点都为“5”，那么返回的结果从位点“5”开始。
   查询结果参数解释

参数	描述
分区	消息的Topic分区。
位点	消息的所在的位点。
Key	消息的键（已强制转化为String类型）。
Value	消息的值（已强制转化为String类型），即消息的具体内容。
消息创建时间	发送消息时，客户端自带的或是您指定的`ProducerRecord`中的消息创建时间。 **说明** ○ 如果配置了该字段，则按配置值显示。 ○ 如果未配置该字段，则默认取消息发送时的系统时间。 ○ 如果显示值为1970/x/x x:x:x，则说明发送时间配置为0或其他有误的值。 ○ 0.9及以前版本的消息队列Kafka版客户端不支持配置该时间。
操作	○ 单击**下载 Key**：下载消息的键值。 ○ 单击**下载 Value**：下载消息的具体内容。 **注意** ○ 查询到的每条消息在控制台上最多显示1 KB的内容，超过1 KB的部分将自动截断。如需查看完整的消息内容，请下载相应的消息。 ○ 仅专业版支持下载消息。 ○ 下载的消息最大为10 MB。如果消息超过10 MB，则只下载10 MB的内容。

3.2.3. 公网

3.2.3.1. 作为Input接入

消息队列Kafka版可以作为Input接入Filebeat。本文说明如何在公网环境下通过Filebeat从消息队列Kafka版消费消息。

背景信息

在开始本教程前，请确保您已完成以下操作：

- 购买并部署消息队列Kafka版实例。具体操作，请参见公网+VPC接入。
- 下载并安装Filebeat。具体操作，请参见Download Filebeat。
- 下载并安装JDK 8。具体操作，请参见Download JDK 8。

步骤一：获取接入点

Filebeat通过消息队列Kafka版的接入点与消息队列Kafka版建立连接。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击作为Input接入Filebeat的实例名称。
4. 在**实例详情**页面的**接入点信息**区域，获取实例的接入点。在**配置信息**区域，获取**用户名**与**密码**。

> **说明** 不同接入点的差异，请参见接入点对比。

步骤二：创建Topic

创建用于存储消息的Topic。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。

> **注意** Topic需要在应用程序所在的地域（即所部署的ECS的所在地域）进行创建。Topic不能跨地域使用。例如Topic创建在华北2（北京）这个地域，那么消息生产端和消费端也必须运行在华北2（北京）的ECS。

3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，单击**创建 Topic**。
6. 在**创建 Topic**面板，设置Topic属性，然后单击**确定**。

参数	说明	示例
名称	Topic名称。	demo
描述	Topic的简单描述。	demo test
分区数	Topic的分区数量。	12
存储引擎	Topic消息的存储引擎。 消息队列Kafka版支持以下两种存储引擎。 ○ **云存储**：底层接入阿里云云盘，具有低时延、高性能、持久性、高可靠等特点，采用分布式3副本机制。实例的**规格类型**为**标准版（高写版）**时，存储引擎只能为**云存储**。 ○ **Local 存储**：使用原生Kafka的ISR复制算法，采用分布式3副本机制。	**云存储**

参数	说明	示例
消息类型	Topic消息的类型。 ◦ **普通消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，可能会造成消息乱序。当**存储引擎**选择**云存储**时，默认选择**普通消息**。 ◦ **分区顺序消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，仍然保证分区内按照发送顺序存储。但是会出现部分分区发送消息失败，等到分区恢复后即可恢复正常。当**存储引擎**选择**Local 存储**时，默认选择**分区顺序消息**。	**普通消息**

参数	说明	示例
日志清理策略	Topic日志的清理策略。 当**存储引擎**选择**Local 存储**时，需要配置**日志清理策略**。 消息队列Kafka版支持以下两种日志清理策略。 ○ **Delete**：默认的消息清理策略。在磁盘容量充足的情况下，保留在最长保留时间范围内的消息；在磁盘容量不足时（一般磁盘使用率超过85%视为不足），将提前删除旧消息，以保证服务可用性。 ○ **Compact**：使用Kafka Log Compaction日志清理策略。Log Compaction清理策略保证相同Key的消息，最新的value值一定会被保留。主要适用于系统宕机后恢复状态，系统重启后重新加载缓存等场景。例如，在使用Kafka Connect或Confluent Schema Registry时，需要使用Kafka Compact Topic存储系统状态信息或配置信息。 **注意** Compact Topic一般只用在某些生态组件中，例如Kafka Connect或Confluent Schema Registry，其他情况的消息收发请勿为Topic设置该属性。具体信息，请参见消息队列Kafka版Demo库。	**Compact**
标签	Topic的标签。	demo

创建完成后，在**Topic 管理**页面的列表中显示已创建的Topic。

步骤三：发送消息

向创建的Topic发送消息。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，找到目标Topic，在其**操作**列中，选择**更多 > 体验发送消息**。
6. 在**快速体验消息收发**面板，发送测试消息。

- **发送方式**选择**控制台**。
  a. 在**消息 Key**文本框中输入消息的Key值，例如demo。
  b. 在**消息内容**文本框输入测试的消息内容，例如 {"key": "test"}。
  c. 设置**发送到指定分区**，选择是否指定分区。
     - 单击**是**，在**分区 ID**文本框中输入分区的ID，例如0。如果您需查询分区的ID，请参见查看分区状态。
     - 单击**否**，不指定分区。
  d. 根据界面提示信息，通过SDK订阅消息，或者执行Docker命令订阅消息。
- **发送方式**选择**Docker**，运行Docker容器。
  a. 执行**运行 Docker 容器生产示例消息**区域的Docker命令，发送消息。
  b. 执行**发送后如何消费消息？**区域的Docker命令，订阅消息。
- **发送方式**选择**SDK**，根据您的业务需求，选择需要的语言或者框架的SDK以及接入方式，通过SDK体验消息收发。

步骤四：创建Group

创建Filebeat所属的Group。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Group 管理**。
5. 在**Group 管理**页面，单击**创建 Group**。
6. 在**创建 Group**面板的**Group ID**文本框输入Group的名称，在**描述**文本框简要描述Group，并给Group添加标签，单击**确定**。
   创建完成后，在**Group 管理**页面的列表中显示已创建的Group。

步骤五：Filebeat消费消息

在安装了Filebeat的机器上启动Filebeat，从创建的Topic中消费消息。

1. 执行**cd**命令切换到Filebeat的安装目录。
2. 执行以下命令下载CA证书文件。

```
wget https://code.aliyun.com/alikafka/aliware-kafka-demos/raw/master/kafka-filebeat-demo/vpc-ssl/ca-cert
```

3. 创建*input.yml*配置文件。
   i. 执行命令 `vim input.yml` 创建空的配置文件。
   ii. 按i键进入插入模式。
   iii. 输入以下内容。

```
filebeat.inputs:
- type: kafka
  hosts:
    - alikafka-pre-cn-zv**********-1.alikafka.aliyuncs.com:9093
    - alikafka-pre-cn-zv**********-2.alikafka.aliyuncs.com:9093
    - alikafka-pre-cn-zv**********-3.alikafka.aliyuncs.com:9093
  username: "alikafka_pre-cn-v641e1dt***"
  password: "aeN3WLRoMPRXmAP2jvJuGk84Kuuo***"
  topics: ["filebeat_test"]
  group_id: "filebeat_group"
  ssl.certificate_authorities: ["/root/filebeat/filebeat-7.7.0-linux-x86_64/ca-cert
"]
  ssl.verification_mode: none
output.console:
  pretty: true
```

参数	描述	示例值
hosts	消息队列Kafka版提供的公网接入点为SSL接入点。	- alikafka-pre-cn-zv**********-1.alikafka.aliyuncs.com:9093 - alikafka-pre-cn-zv**********-2.alikafka.aliyuncs.com:9093 - alikafka-pre-cn-zv**********-3.alikafka.aliyuncs.com:9093
username	公网/VPC实例的用户名。	alikafka_pre-cn-v641e1d***
password	公网/VPC实例的密码。	aeN3WLRoMPRXmAP2jvJuGk84Kuuo***
topics	Topic的名称。	filebeat_test
group_id	Group的名称。	filebeat_group
ssl.certificate_authorities	CA证书所在位置。	/root/filebeat/filebeat-7.7.0-linux-x86_64/ca-cert
ssl.verification_mode	认证模式。	none

更多参数设置，请参见Kafka input plugin。

iv. 按*Esc*键回到命令行模式。

v. 按*：*键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

4. 执行以下命令消费消息。

```
./filebeat -c ./input.yml
```

3.2.3.2. 作为Output接入

消息队列Kafka版可以作为Output接入Filebeat。本文说明如何在公网环境下通过Filebeat向消息队列Kafka版发送消息。

前提条件

在开始本教程前，请确保您已完成以下操作：

- 购买并部署消息队列Kafka版实例。具体操作，请参见公网+VPC接入。
- 下载并安装Filebeat。具体操作，请参见Download Filebeat。
- 下载并安装JDK 8。具体操作，请参见Download JDK 8。

步骤一：获取接入点与用户名密码

Filebeat通过消息队列Kafka版的接入点与消息队列Kafka版建立连接。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。

3. 在**实例列表**页面，单击作为Output接入Filebeat的实例的名称。
4. 在**实例详情**页面的**接入点信息**区域，获取实例的接入点。在**配置信息**区域，获取**用户名**与**密码**。

> **说明** 不同接入点的差异，请参见接入点对比。

步骤二：创建Topic

创建用于存储消息的Topic。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。

> **注意** Topic需要在应用程序所在的地域（即所部署的ECS的所在地域）进行创建。Topic不能跨地域使用。例如Topic创建在华北2（北京）这个地域，那么消息生产端和消费端也必须运行在华北2（北京）的ECS。

3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，单击**创建 Topic**。
6. 在**创建 Topic**面板，设置Topic属性，然后单击**确定**。

参数	说明	示例
名称	Topic名称。	demo
描述	Topic的简单描述。	demo test
分区数	Topic的分区数量。	12
存储引擎	Topic消息的存储引擎。 消息队列Kafka版支持以下两种存储引擎。 ○ **云存储**：底层接入阿里云云盘，具有低时延、高性能、持久性、高可靠等特点，采用分布式3副本机制。实例的**规格类型**为**标准版（高写版）**时，存储引擎只能为**云存储**。 ○ **Local 存储**：使用原生Kafka的ISR复制算法，采用分布式3副本机制。	**云存储**

参数	说明	示例
消息类型	Topic消息的类型。 ○ **普通消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，可能会造成消息乱序。当**存储引擎**选择**云存储**时，默认选择**普通消息**。 ○ **分区顺序消息**：默认情况下，保证相同Key的消息分布在同一个分区中，且分区内消息按照发送顺序存储。集群中出现机器宕机时，仍然保证分区内按照发送顺序存储。但是会出现部分分区发送消息失败，等到分区恢复后即可恢复正常。当**存储引擎**选择**Local 存储**时，默认选择**分区顺序消息**。	**普通消息**

参数	说明	示例
日志清理策略	Topic日志的清理策略。 当**存储引擎**选择**Local 存储**时，需要配置**日志清理策略**。 消息队列Kafka版支持以下两种日志清理策略。 ○ **Delete**：默认的消息清理策略。在磁盘容量充足的情况下，保留在最长保留时间范围内的消息；在磁盘容量不足时（一般磁盘使用率超过85%视为不足），将提前删除旧消息，以保证服务可用性。 ○ **Compact**：使用Kafka Log Compaction日志清理策略。Log Compaction清理策略保证相同Key的消息，最新的value值一定会被保留。主要适用于系统宕机后恢复状态，系统重启后重新加载缓存等场景。例如，在使用Kafka Connect或Confluent Schema Registry时，需要使用Kafka Compact Topic存储系统状态信息或配置信息。 **注意** Compact Topic一般只用在某些生态组件中，例如Kafka Connect或Confluent Schema Registry，其他情况的消息收发请勿为Topic设置该属性。具体信息，请参见消息队列Kafka版Demo库。	**Compact**
标签	Topic的标签。	demo

创建完成后，在**Topic 管理**页面的列表中显示已创建的Topic。

步骤三：Filebeat发送消息

在安装了Filebeat的机器上启动Filebeat，向创建的Topic发送消息。

1. 执行**cd**命令切换到Filebeat的安装目录。
2. 执行以下命令下载CA证书文件。

```
wget https://code.aliyun.com/alikafka/aliware-kafka-demos/raw/master/kafka-filebeat-demo/vpc-ssl/ca-cert
```

3. 创建*output.conf*配置文件。
   i. 执行命令 `vim output.conf` 创建空的配置文件。
   ii. 按键进入插入模式。
   iii. 输入以下内容。

```
filebeat.inputs:
- type: stdin
output.kafka:
  hosts: ["alikafka-pre-cn-zv**********-1.alikafka.aliyuncs.com:9093", "alikafka-pre-cn-zv**********-2.alikafka.aliyuncs.com:9093", "alikafka-pre-cn-zv**********-3.alikafka.aliyuncs.com:9093"]
  username: "alikafka_pre-cn-v641e1d***"
  password: "aeN3WLRoMPRXmAP2jvJuGk84Kuuo***"
  topic: 'filebeat_test'
  partition.round_robin:
    reachable_only: false
  ssl.certificate_authorities: ["/root/filebeat/filebeat-7.7.0-linux-x86_64/tasks/vpc_ssl/ca-cert"]
  ssl.verification_mode: none
  required_acks: 1
  compression: none
  max_message_bytes: 1000000
```

参数	描述	示例值
hosts	消息队列Kafka版提供的公网接入点为SSL接入点。	alikafka-pre-cn-zv**********-1.alikafka.aliyuncs.com:9093，alikafka-pre-cn-zv**********-2.alikafka.aliyuncs.com:9093，alikafka-pre-cn-zv**********-3.alikafka.aliyuncs.com:9093
username	公网/VPC实例的用户名。	alikafka_pre-cn-v641e1d***
password	公网/VPC实例的密码。	aeN3WLRoMPRXmAP2jvJuGk84Kuuo***
topic	Topic的名称。	filebeat_test
reachable_only	消息是否只发送到可用的分区。取值： ■ true：如果主分区不可用，输出可能阻塞。 ■ false：即使主分区不可用，输出不被阻塞。	false
ssl.certificate_authorities	CA证书所在位置。	/root/filebeat/filebeat-7.7.0-linux-x86_64/ca-cert

参数	描述	示例值
ssl.verification_mode	认证模式。	none
required_acks	ACK可靠性。取值： ■ 0：无响应 ■ 1：等待本地提交 ■ -1：等待所有副本提交 默认值为1。	1
compression	数据压缩编译码器。默认值为gzip。取值： ■ none：无 ■ snappy：用来压缩和解压缩的C++开发包 ■ lz4：着重于压缩和解压缩速度的无损数据压缩算法 ■ gzip：GNU自由软件的文件压缩程序	none
max_message_bytes	最大消息大小。单位为字节。默认值为1000000。该值应小于您配置的消息队列Kafka版最大消息大小。	1000000

更多参数说明，请参见Kafka output plugin。

iv. 按*Esc*键回到命令行模式。

v. 按*：*键进入底行模式，输入*wq*，然后按回车键保存文件并退出。

4. 向创建的Topic发送消息。

i. 执行 `./filebeat -c ./output.yml` 。

ii. 输入*test*，然后按回车键。

步骤四：查看Topic分区

查看消息发送到Topic的情况。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**Topic 管理**。
5. 在**Topic 管理**页面，找到目标Topic，在其**操作**列中，选择**更多 > 分区状态**。

分区状态信息

参数	说明
分区ID	该Topic分区的ID号。

参数	说明
最小位点	该Topic在当前分区下的最小消费位点。
最大位点	该Topic在当前分区下的最大消费位点。
最近更新时间	本分区中最近一条消息的存储时间。

步骤五：按位点查询消息

您可以根据发送的消息的分区ID和位点信息查询该消息。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 在左侧导航栏，单击**消息查询**。
5. 在**消息查询**页面的**查询方式**列表中，选择**按位点查询**。
6. 在**Topic**列表中，选择消息所属Topic名称；在**分区**列表中，选择消息所属的分区；在**起始位点**文本框，输入消息所在分区的位点，然后单击**查询**。
   展示该查询位点及以后连续的消息。例如，指定的分区和位点都为“5”，那么返回的结果从位点“5”开始。
   查询结果参数解释

参数	描述
分区	消息的Topic分区。
位点	消息的所在的位点。
Key	消息的键（已强制转化为String类型）。
Value	消息的值（已强制转化为String类型），即消息的具体内容。

参数	描述
消息创建时间	发送消息时，客户端自带的或是您指定的`ProducerRecord`中的消息创建时间。 **说明** ○ 如果配置了该字段，则按配置值显示。 ○ 如果未配置该字段，则默认取消息发送时的系统时间。 ○ 如果显示值为1970/x/x x:x:x，则说明发送时间配置为0或其他有误的值。 ○ 0.9及以前版本的消息队列Kafka版客户端不支持配置该时间。
操作	○ 单击**下载 Key**：下载消息的键值。 ○ 单击**下载 Value**：下载消息的具体内容。 **注意** ○ 查询到的每条消息在控制台上最多显示1 KB的内容，超过1 KB的部分将自动截断。如需查看完整的消息内容，请下载相应的消息。 ○ 仅专业版支持下载消息。 ○ 下载的消息最大为10 MB。如果消息超过10 MB，则只下载10 MB的内容。

3.2.4. Filebeat发送失败问题

问题现象

在消息队列Kafka版控制台的**Topic 管理**页面，您已成功创建Topic，但使用Filebeat向该Topic发送消息却出现`Request was for a topic or partition that does not exist on this broker.`的错误提示。

可能原因

Filebeat向Topic发送消息时，会自动将Topic名称中的大写字母转换成小写字母。例如，您使用Filebeat向名称为AAA的Topic发送消息时，Filebeat会自动转换成向名称为aaa的Topic发送消息。由于发送消息的请求无法找到订阅的Topic，所以出现`Request was for a topic or partition that does not exist on this broker.`的错误提示。

解决方案

操作步骤

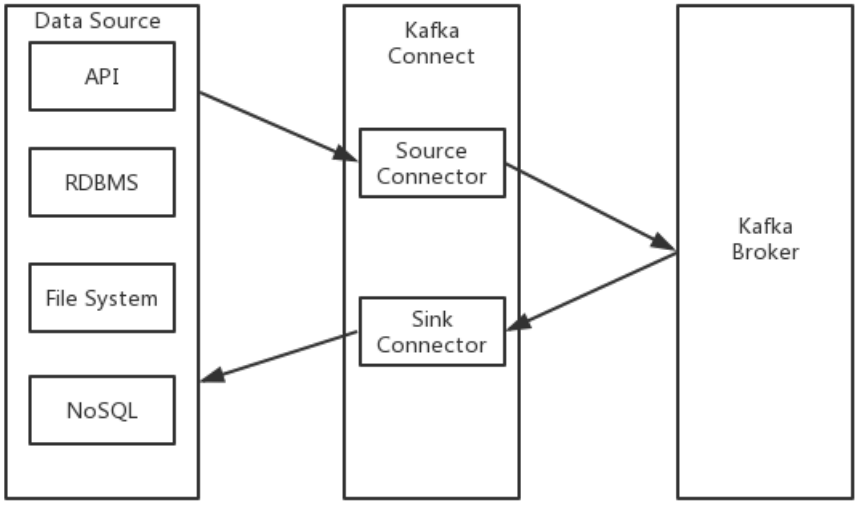
1. 在消息队列Kafka版控制台的**Topic 管理**页面创建Topic，请使用全小写字母给Topic命名。

3.3. 使用Kafka Connect将MySQL数据同步至消息队列Kafka版

本教程介绍如何使用Kafka Connect的Source Connector将MySQL的数据同步至消息队列Kafka版。

背景信息

Kafka Connect主要用于将数据流输入和输出消息队列Kafka版。Kafka Connect主要通过各种Source Connector的实现，将数据从第三方系统输入到Kafka Broker，通过各种Sink Connector实现，将数据从Kafka Broker中导入到第三方系统。



前提条件

在开始本教程前，请确保您已完成以下操作：

- 下载MySQL Source Connector。

  > **说明** 本教程以0.5.2版本的MySQL Source Connector为例。

- 下载Kafka Connect。

  > **说明** 本教程以0.10.2.2版本的Kafka Connect为例。

- 安装Docker。

步骤一：配置Kafka Connect

1. 将下载完成的MySQL Connector解压到指定目录。
2. 在Kafka Connect的配置文件*connect-distributed.properties*中配置插件安装位置。

```
plugin.path=/kafka/connect/plugins
```

> **注意**
> Kafka Connect的早期版本不支持配置*plugin.path*，您需要在*CLASSPATH*中指定插件位置。
>
> ```
> export CLASSPATH=/kafka/connect/plugins/mysql-connector/*
> ```

步骤二：启动Kafka Connect

在配置好*connect-distributed.properties*后，执行以下命令启动Kafka Connect。

- 公网接入
  i. 执行命令`export KAFKA_OPTS="-Djava.security.auth.login.config=kafka_client_jaas.conf"`设置*java.security.auth.login.config*。
  ii. 执行命令`bin/connect-distributed.sh config/connect-distributed.properties`启动Kafka Connect。
- VPC接入
  执行命令`bin/connect-distributed.sh config/connect-distributed.properties`启动Kafka Connect。

步骤三：安装MySQL

1. 下载docker-compose-mysql.yaml。
2. 执行以下命令安装MySQL。

```
export DEBEZIUM_VERSION=0.5
docker-compose -f docker-compose-mysql.yaml up
```

步骤四：配置MySQL

1. 执行以下命令开启MySQL的binlog写入功能，并配置binlog模式为row。

```
[mysqld]
log-bin=mysql-bin
binlog-format=ROW
server_id=1
```

2. 执行以下命令设置MySQL的User权限。

```
GRANT SELECT, RELOAD, SHOW DATABASES, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO '
debezium' IDENTIFIED BY 'dbz';
```

> **说明** 示例中MySQL的User为*debezium*，密码为*dbz*。

步骤五：启动MySQL Connector

1. 下载register-mysql.json。
2. 编辑*register-mysql.json*。
   - VPC接入

```
## 消息队列Kafka版接入点，通过控制台获取。
## 您在控制台获取的默认接入点。
"database.history.kafka.bootstrap.servers" : "kafka:9092",
## 需要提前在控制台创建同名Topic，在本例中创建Topic：server1。
## 所有Table的变更数据，会记录在server1.$DATABASE.$TABLE的Topic中，如 server1.inventory.products。
## 因此用户需要提前在控制台中创建所有相关Topic。
"database.server.name": "server1",
## 记录schema变化信息将记录在这个Topic中。
## 需要提前在控制台创建。
"database.history.kafka.topic": "schema-changes-inventory"
```

- 公网接入

```
## 消息队列Kafka版接入点，通过控制台获取。存储db中schema变化信息。
## 您在控制台获取的SSL接入点。
"database.history.kafka.bootstrap.servers" : "kafka:9092",
## 需要提前在控制台创建同名Topic，在本例中创建Topic：server1。
## 所有Table的变更数据，会记录在server1.$DATABASE.$TABLE的Topic中，如 server1.testDB.products。
## 因此用户需要提前在控制台中创建所有相关Topic。
"database.server.name": "server1",
## schema变化信息将记录在这个Topic中。
## 需要提前在控制台创建。
"database.history.kafka.topic": "schema-changes-inventory",
## SSL公网方式访问配置。
"database.history.producer.ssl.truststore.location": "kafka.client.truststore.jks",
"database.history.producer.ssl.truststore.password": "KafkaOnsClient",
"database.history.producer.security.protocol": "SASL_SSL",
"database.history.producer.sasl.mechanism": "PLAIN",
"database.history.consumer.ssl.truststore.location": "kafka.client.truststore.jks",
"database.history.consumer.ssl.truststore.password": "KafkaOnsClient",
"database.history.consumer.security.protocol": "SASL_SSL",
"database.history.consumer.sasl.mechanism": "PLAIN",
```

3. 配置好register-mysql.json后，您需要根据配置在控制台创建相应的Topic，相关操作步骤，请参见步骤一：创建Topic。

   按照本教程中的方式安装的MySQL，您可以看到MySQL中已经提前创建好了*database：inventory*。其中有四张表：

   - *customers*
   - *orders*
   - *products*
   - *products_on_hand*

   根据以上配置，您需要使用OpenAPI创建Topic：

   - *server1*
   - *server1.inventory.customers*
   - *server1.inventory.orders*
   - *server1.inventory.products*
   - *server1.inventory.products_on_hand*

在*register-mysql.json*中，配置了将schema变化信息记录在*schema-changes-testDB*，因此您还需要使用OpenAPI创建Topic：*schema-changes-inventory*。使用OpenAPI创建Topic，请参见CreateTopic。

4. 执行以下命令启动MySQL Connector。

```
curl -i -X POST -H "Accept:application/json" -H  "Content-Type:application/json" http:/
/localhost:8083/connectors/ -d @register-mysql.json
```

结果验证

按照以下步骤操作确认消息队列Kafka版能否接收到MySQL的变更数据。

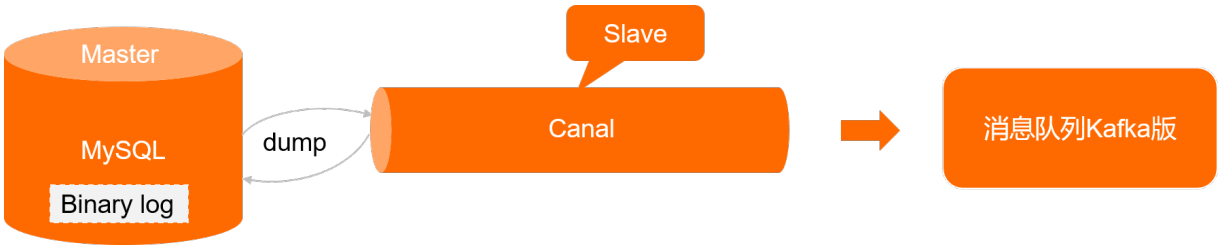
1. 变更MySQL Table中的数据。
2. 在控制台的**消息查询**页面，查询变更数据。

3.4. 使用Canal将MySQL的数据同步至消息队列Kafka版

本教程介绍如何使用Canal将MySQL的数据同步至消息队列Kafka版。

背景信息

Canal的主要用途是基于MySQL数据库增量日志解析，提供增量数据订阅和消费。Canal伪装自己为MySQL Slave，向MySQL Master发送dump请求。MySQL Master收到dump请求，开始推送Binary log给Canal，Canal解析Binary log来同步数据。Canal与消息队列Kafka版建立对接，您可以把MySQL更新的数据写入到消息队列Kafka版中来分析。其详细的工作原理，请参见Canal官网。



前提条件

在开始本教程前，请确保您已完成以下操作：

- 安装MySQL，并进行相关初始化与设置。具体操作，请参见 Canal QuickStart。
- 在消息队列Kafka版控制台创建实例以及Topic资源。具体操作，请参见步骤三：创建资源。

操作步骤

1. 下载Canal压缩包，本教程以1.1.5版本为例。
2. 执行以下命令，创建目录文件夹。本教程以*/home/doc/tools/canal.deployer-1.1.5*路径为例。

```
mkdir -p /home/doc/tools/canal.deployer-1.1.5
```

3. 将Canal压缩包复制到*/home/doc/tools/canal.deployer-1.1.5*路径并解压。

```
tar -zxvf canal.deployer-1.1.5-SNAPSHOT.tar.gz -C /home/doc/tools/canal.deployer-1.1.5
```

4. 在*/home/doc/tools/canal.deployer-1.1.5*路径，执行以下命令，编辑*instance.properties*文件。

```
vi conf/example/instance.properties
```

根据instance.properties参数列表配置参数。

```
#  根据实际情况修改为您的数据库信息。
#################################################
...
# 数据库地址。
canal.instance.master.address=192.168.XX.XX:3306
# username/password为数据库的用户名和密码。
...
canal.instance.dbUsername=****
canal.instance.dbPassword=****
...
# mq config
# 您在消息队列Kafka版控制台创建的Topic。
canal.mq.topic=mysql_test
# 针对数据库名或者表名发送动态Topic。
#canal.mq.dynamicTopic=mytest,.*,mytest.user,mytest\\..*,.*\\..*
# 数据同步到消息队列Kafka版Topic的指定分区。
canal.mq.partition=0
# 以下两个参数配置与canal.mq.partition互斥。配置以下两个参数可以使数据发送至消息队列Kafka版Topic的不同分区。
#canal.mq.partitionsNum=3
#库名.表名：唯一主键，多个表之间用逗号分隔。
#canal.mq.partitionHash=mytest.person:id,mytest.role:id
#################################################
```

*instance.properties*参数列表

参数	是否必选	描述
canal.instance.master.address	是	MySQL数据库的连接地址。
canal.instance.dbUsername	是	MySQL数据库的用户名。
canal.instance.dbPassword	是	MySQL数据库的用户名密码。
canal.mq.topic	是	消息队列Kafka版实例的Topic。您可以在消息队列Kafka版控制台的**Topic 管理**页面创建。具体操作，请参见步骤三：创建资源。
canal.mq.dynamicTopic	否	动态Topic规则表达式。设置Topic匹配规则表达式，可以将不同的数据表数据同步至不同的Topic。具体设置方法，请参见参数说明。
canal.mq.partition	否	数据库数据同步到消息队列Kafka版Topic的指定分区。
canal.mq.partitionsNum	否	Topic的分区数量。该参数与canal.mq.partitionHash一起使用，可以将数据同步至消息队列Kafka版Topic不同的分区。

参数	是否必选	描述
canal.mq.partitionHash	否	分区的规则表达式。具体设置方法，请参见参数说明。

5. 执行以下命令，编辑*canal.properties*文件。

```
vi conf/canal.properties
```

根据canal.properties参数列表说明配置参数。

- 公网环境，消息采用SASL_SSL协议进行鉴权并加密，通过SSL接入点访问消息队列Kafka版。接入点的详细信息，请参见接入点对比。

```
# ...
# 您需设置为kafka。
canal.serverMode = kafka
# ...
# kafka配置。
#在消息队列Kafka版实例详情页面获取的SSL接入点。
kafka.bootstrap.servers = alikafka-pre-cn-zv**********-1.alikafka.aliyuncs.com:9093,alikafka-pre-cn-zv**********-2.alikafka.aliyuncs.com:9093,alikafka-pre-cn-zv**********-3.alikafka.aliyuncs.com:9093
# 参数的默认设置如下所示，您可以根据实际情况调整。
kafka.acks = all
kafka.compression.type = none
kafka.batch.size = 16384
kafka.linger.ms = 1
kafka.max.request.size = 1048576
kafka.buffer.memory = 33554432
kafka.max.in.flight.requests.per.connection = 1
kafka.retries = 0
# 公网环境，通过SASL_SSL鉴权并加密，您需配置网络协议与身份校验机制。
kafka.ssl.truststore.location= ../conf/kafka_client_truststore_jks
kafka.ssl.truststore.password= KafkaOnsClient
kafka.security.protocol= SASL_SSL
kafka.sasl.mechanism = PLAIN
kafka.ssl.endpoint.identification.algorithm =
```

*canal.properties*参数列表

参数	是否必选	描述
canal.serverMode	是	您需设置为*kafka*。
kafka.bootstrap.servers	是	消息队列Kafka版实例接入点。您可在消息队列Kafka版控制台的**实例详情**页面的**接入点信息**区域获取。

参数	是否必选	描述
kafka.ssl.truststore.location	是	SSL根证书kafka.client.truststore.jks的存放路径。 **说明** 公网环境下，消息必须进行鉴权与加密，才能确保传输的安全。即需通过SSL接入点采用SASL_SSL协议进行传输。具体信息，请参见接入点对比。
kafka.acks	是	消息队列Kafka版接收到数据之后给客户端发出的确认信号。取值说明如下： ■ *0*：表示客户端不需要等待任何确认收到的信息。 ■ *1*：表示等待Leader成功写入而不等待所有备份是否成功写入。 ■ *all*：表示等待Leader成功写入并且所有备份都成功写入。
kafka.compression.type	是	压缩数据的压缩算法，默认是无压缩。取值如下： ■ *none*。 ■ *gzip*。 ■ *snappy*。

参数	是否必选	描述
kafka.batch.size	是	客户端数据块攒批的大小。单位：Byte。 该参数控制批量处理消息的字节数。客户端发送到Brokers的请求将包含多个批量处理，以减少请求次数。较小的批量处理数值可能降低吞吐量，而较大的批量处理数值将会浪费更多内存空间，分配一个指定的批量处理消息缓冲区有助于提高客户端和服务端的性能。 **说明** kafka.batch.size与kafka.linger.ms参数都是控制批量处理消息的条件，满足其中一个参数的条件，客户端就攒批完成，消息进入待发送状态。
kafka.linger.ms	是	客户端数据块攒批的最大时长。单位：ms。 客户端设定攒批消息的延迟时间，以批量处理消息，减少请求次数。
kafka.max.request.size	是	客户端每次请求的最大字节数。
kafka.buffer.memory	是	缓存数据的内存大小。
kafka.max.in.flight.requests.per.connection	是	限制客户端在单个连接上能够发送的未响应请求的个数。设置此值是*1*表示Broker在响应请求之前客户端不能再向同一个Broker发送请求。
kafka.retries	是	消息发送失败时，是否重复发送。设置为*0*，表示不会重复发送；设置大于0的值，客户端重新发送数据。
kafka.ssl.truststore.password	是	SSL根证书的密码，设置为*KafkaOnsClient*。
kafka.security.protocol	是	采用SASL_SSL协议进行鉴权并加密，即设置为*SASL_SSL*。

参数	是否必选	描述
kafka.sasl.mechanism	是	SASL身份认证的机制。SSL接入点采用*PLAIN*机制验证身份。

公网环境，需通过SASL进行身份校验，需要在*bin/startup.sh*配置环境变量，并编辑*kafka_client_producer_jaas.conf*文件，配置消息队列Kafka版实例的用户名与密码。

a. 执行 `vi bin/startup.sh` 命令，编辑*startup.sh*文件，配置环境变量。

```
JAVA_OPTS=" $JAVA_OPTS -Djava.awt.headless=true -Djava.net.preferIPv4Stack=true -Dfile.encoding=UTF-8 -Djava.security.auth.login.config=/home/doc/tools/canal.deployer-1.1.5/conf/kafka_client_jaas.conf"
```

b. 执行 `vi conf/kafka_client_producer_jaas.conf` 命令，编辑*kafka_client_producer_jaas.conf*文件，配置实例用户名与密码信息。

> **说明**
>
> - 如果实例未开启ACL，您可以在消息队列Kafka版控制台的**实例详情**页面获取默认用户的用户名和密码。
> - 如果实例已开启ACL，请确保要使用的SASL用户为PLAIN类型且已授权收发消息的权限。具体信息，请参见SASL用户授权。

```
KafkaClient {  org.apache.kafka.common.security.plain.PlainLoginModule required
               username="实例的用户名"
               password="实例的用户名密码";
};
```

- VPC环境，消息采用PLAINTEXT协议不鉴权不加密传输，通过默认接入点访问消息队列Kafka版，仅需配置canal.serverMode与kafka.bootstrap.servers参数。接入点的详细信息，请参见接入点对比。

```
# ...
# 您需设置为kafka。
canal.serverMode = kafka
# ...
# kafka配置。
# 在消息队列Kafka版实例详情页面获取的默认接入点。
kafka.bootstrap.servers = alikafka-pre-cn-zv**********-1-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-2-vpc.alikafka.aliyuncs.com:9092,alikafka-pre-cn-zv**********-3-vpc.alikafka.aliyuncs.com:9092
# 以下参数请您可以按照实际情况调整，也可以保持默认设置。
kafka.acks = all
kafka.compression.type = none
kafka.batch.size = 16384
kafka.linger.ms = 1
kafka.max.request.size = 1048576
kafka.buffer.memory = 33554432
kafka.max.in.flight.requests.per.connection = 1
kafka.retries = 0
```

6. 在*/home/doc/tools/canal.deployer-1.1.5*路径，执行以下命令，启动Canal。

```
sh bin/startup.sh
```

- 查看*/home/doc/tools/canal.deployer-1.1.5/logs/canal/canal.log*日志文件，确认Canal与消息队列Kafka版连接成功，Canal正在运行。

```
2013-02-05 22:45:27.967 [main] INFO  com.alibaba.otter.canal.deployer.CanalLauncher -
## start the canal server.
2013-02-05 22:45:28.113 [main] INFO  com.alibaba.otter.canal.deployer.CanalController
- ## start the canal server[10.1.XX.XX:11111]
2013-02-05 22:45:28.210 [main] INFO  com.alibaba.otter.canal.deployer.CanalLauncher -
## the canal server is running now ......
```

- 查看*/home/doc/tools/canal.deployer-1.1.5/logs/example/example.log*日志文件，确认Canal Instance已启动。

```
2013-02-05 22:50:45.636 [main] INFO  c.a.o.c.i.spring.support.PropertyPlaceholderConf
igurer - Loading properties file from class path resource [canal.properties]
2013-02-05 22:50:45.641 [main] INFO  c.a.o.c.i.spring.support.PropertyPlaceholderConf
igurer - Loading properties file from class path resource [example/instance.propertie
s]
2013-02-05 22:50:45.803 [main] INFO  c.a.otter.canal.instance.spring.CanalInstanceWit
hSpring - start CannalInstance for 1-example
2013-02-05 22:50:45.810 [main] INFO  c.a.otter.canal.instance.spring.CanalInstanceWit
hSpring - start successful....
```

测试验证

启动Canal之后，进行数据同步验证。

1. 在MySQL数据库，新建数据表T_Student。数据表数据示例如下：

```
mysql> select * from T_Student;
+--------+---------+------+------+
| stuNum | stuName | age  | sex  |
+--------+---------+------+------+
|      1 | 小王    |   18 | girl |
|      2 | 小张    |   17 | boy  |
+--------+---------+------+------+
2 rows in set (0.00 sec)
```

查看*/home/doc/tools/canal.deployer-1.1.5/logs/example/meta.log*日志文件，数据库的每次增删改操作，都会在*meta.log*中生成一条记录，查看该日志可以确认Canal是否有采集到数据。

```
tail -f example/meta.log
2020-07-29 09:21:05.110 - clientId:1001 cursor:[log.000001,29723,1591190230000,1,] addr
ess[/192.168.XX.XX:3306]
2020-07-29 09:23:46.109 - clientId:1001 cursor:[log.000001,30047,1595985825000,1,] addr
ess[localhost/192.168.XX.XX:3306]
2020-07-29 09:24:50.547 - clientId:1001 cursor:[log.000001,30047,1595985825000,1,] addr
ess[/192.168.XX.XX:3306]
2020-07-29 09:26:45.547 - clientId:1001 cursor:[log.000001,30143,1595986005000,1,] addr
ess[localhost/192.168.XX.XX:3306]
2020-07-29 09:30:04.546 - clientId:1001 cursor:[log.000001,30467,1595986204000,1,] addr
ess[localhost/192.168.XX.XX:3306]
2020-07-29 09:30:16.546 - clientId:1001 cursor:[log.000001,30734,1595986215000,1,] addr
ess[localhost/192.168.XX.XX:3306]
2020-07-29 09:30:36.547 - clientId:1001 cursor:[log.000001,31001,1595986236000,1,] addr
ess[localhost/192.168.XX.XX:3306]
```

2. 登录消息队列Kafka版控制台，查询消息，确认MySQL的数据被同步在消息队列Kafka版。控制台查询消息的具体操作，请参见查询消息。

3. 数据同步完毕，执行以下命令，关闭Canal。

```
sh bin/stop.sh
```

3.5. 使用Kafka Connect将SQL Server数据同步至消息队列Kafka版

本教程介绍如何使用Kafka Connect的Source Connector将SQL Server的数据同步至消息队列Kafka版。

前提条件

在开始本教程前，请确保您已完成以下操作：

- 已下载SQL Server Source Connector。具体信息，请参见SQL Server Source Connector。
- 已下载Kafka Connect。具体信息，请参见Kafka Connect。

> **说明** SQL Server Source Connector目前只支持2.1.0及以上版本的Kafka Connect。

- 已下载Docker。具体信息，请参见Docker。

步骤一：配置Kafka Connect

1. 将下载完成的SQL Server Connector解压到指定目录。
2. 在Kafka Connect的配置文件*connect-distributed.properties*中配置插件安装位置。

```
## 指定插件解压后的路径。
plugin.path=/kafka/connect/plugins
```

> 注意
> Kafka Connect的早期版本不支持配置*plugin.path*，您需要在*CLASSPATH*中指定插件位置。

```
export CLASSPATH=/kafka/connect/plugins/sqlserver-connector/*
```

步骤二：启动Kafka Connect

配置好*connect-distributed.properties*后，执行以下命令启动Kafka Connect。

1. 如果是公网接入，需先设置java.security.auth.login.config，如果是VPC接入，可以跳过这一步。

```
export KAFKA_OPTS="-Djava.security.auth.login.config=kafka_client_jaas.conf"
```

2. 启动Kafka Connect。

```
bin/connect-distributed.sh config/connect-distributed.properties
```

步骤三：安装SQL Server

> 注意 SQL Server 2016 SP1以上版本支持CDC，因此您的SQL Server版本必须高于该版本。

1. 下载docker-compose-sqlserver.yaml。
2. 执行以下命令安装SQL Server。

```
docker-compose -f docker-compose-sqlserver.yaml up
```

步骤四：配置SQL Server

1. 下载inventory.sql。
2. 执行以下命令初始化SQL Server中的测试数据。

```
cat inventory.sql | docker exec -i tutorial_sqlserver_1 bash -c '/opt/mssql-tools/bin/sqlcmd -U sa -P $SA_PASSWORD'
```

3. （可选）如果您需要监听SQL Server中已有的数据表，请完成以下配置：
   i. 执行以下命令开启CDC配置。

```
## 开启CDC模板数据库。
USE testDB
GO
EXEC sys.sp_cdc_enable_db
GO
```

ii. 执行以下命令开启指定Table的CDC配置。

```
## 开启指定Table的CDC配置。
USE testDB
GO
EXEC sys.sp_cdc_enable_table
@source_schema = N'dbo',
@source_name   = N'MyTable',
@role_name     = N'MyRole',
@filegroup_name = N'MyDB_CT',
@supports_net_changes = 1
GO
```

iii. 执行以下命令确认是否有权限访问CDC Table。

```
EXEC sys.sp_cdc_help_change_data_capture
GO
```

> **说明** 如果返回结果为空，您需要确认是否有权限访问该表。

iv. 执行以下命令确认SQL Server Agent已开启。

```
EXEC master.dbo.xp_servicecontrol N'QUERYSTATE',N'SQLSERVERAGENT'
```

> **说明** 如果返回结果为 `Running`，则说明SQL Server Agent已开启。

步骤五：启动SQL Server Connector

1. 下载register-sqlserver.json。
2. 编辑*register-sqlserver.json*。
   - VPC接入

     ```
     ## 消息队列Kafka版实例的默认接入点，您可以在消息队列Kafka版控制台获取。
     "database.history.kafka.bootstrap.servers" : "kafka:9092",
     ## 您需要提前在消息队列Kafka版控制台创建同名Topic，在本例中创建topic：server1。
     ## 所有table的变更数据，会记录在server1.$DATABASE.$TABLE的topic中，例如server1.testDB.products。
     ## 因此您需要提前在消息队列Kafka版控制台中创建所有相关Topic。
     "database.server.name": "server1",
     ## 记录schema变化信息将记录在该Topic中。
     ## 您需要提前在消息队列Kafka版控制台创建该Topic。
     "database.history.kafka.topic": "schema-changes-inventory"
     ```

   - 公网接入

```
## 消息队列Kafka版实例的SSL接入点，您可以在消息队列Kafka版控制台获取。
"database.history.kafka.bootstrap.servers" : "kafka:9092",
## 您需要提前在消息队列Kafka版控制台创建同名Topic，在本例中创建topic：server1。
## 所有table的变更数据，会记录在server1.$DATABASE.$TABLE的Topic中，例如server1.testDB.products。
## 因此您需要提前在消息队列Kafka版控制台中创建所有相关Topic。
"database.server.name": "server1",
## 记录schema变化信息将记录在该Topic中。
## 您需要提前在消息队列Kafka版控制台创建该Topic。
"database.history.kafka.topic": "schema-changes-inventory",
## 通过SSL接入点访问，还需要修改以下配置。
"database.history.producer.ssl.truststore.location": "kafka.client.truststore.jks",
"database.history.producer.ssl.truststore.password": "KafkaOnsClient",
"database.history.producer.security.protocol": "SASL_SSL",
"database.history.producer.sasl.mechanism": "PLAIN",
"database.history.consumer.ssl.truststore.location": "kafka.client.truststore.jks",
"database.history.consumer.ssl.truststore.password": "KafkaOnsClient",
"database.history.consumer.security.protocol": "SASL_SSL",
"database.history.consumer.sasl.mechanism": "PLAIN",
```

3. 完成*register-sqlserver.json*配置后，您需要根据配置在控制台创建相应的Topic，相关操作步骤请参见步骤一：创建Topic。

   按照本教程中的方式安装的SQL Server，您可以看到SQL Server中已经提前创建*db name：testDB*。其中有四张表：

   - *customers*
   - *orders*
   - *products*
   - *products_on_hand*

   根据以上*register-sqlserver.json*的配置，您需要使用OpenAPI创建Topic：

   - *server1*
   - *server1.testDB.customers*
   - *server1.testDB.orders*
   - *server1.testDB.products*
   - *server1.testDB.products_on_hand*

   在*register-sqlserver.json*中，配置了将schema变化信息记录在*schema-changes-testDB*，因此您还需要使用OpenAPI创建Topic：*schema-changes-inventory*，相关操作请参见CreateTopic。

4. 执行以下命令启动SQL Server。

```
curl -i -X POST -H "Accept:application/json" -H  "Content-Type:application/json" http://localhost:8083/connectors/ -d @register-sqlserver.json
```

结果验证

确认消息队列Kafka版能否接收到SQL Server的变更数据：

1. 变更监听SQL Server中的数据。
2. 在控制台的**消息查询**页面，查询变更消息。具体操作步骤，请参见查询消息。